

PRIMO FÜZETER

STARTER



RAMOSZTÓ-SZOFTVER

**MTA-SZTAKI COSY MŰSZAKI FEJLESZTŐ LEÁNYVÁLLALAT
BUDAPEST 1985**

Szerzők

**TISZAI TAMÁS
TICK JÓZSEF**

Szerkesztette

MAJTÉNYI LÁSZLÓ

Lektorálta

Dr. CSABA LÁSZLÓ

Kiadja

az MTA–SZTAKI COSY Műszaki Fejlesztő Leányvállalat

A változtatás jogát fenntartjuk.

Felelős kiadó

MÓRICZ SÁNDOR
igazgató

Megjelent a GRAFO KIADÓI IRODA gondozásában
5000 példányban, 31,4 (A/5) ív terjedelemben
255—1/85

VTV REPROTECHNIKA

Előszó

A PRIMO otthoni számítógép működtető programjának leírása a "PRIMO füzetek" sorozat keretében jelenik meg. A leírás célja, hogy feltárja a PRIMO számítógép ROM-ban elhelyezkedő rendszerprogramjának működését. Olyan segédeszköz legyen, melynek megismerésével és használatával a szakember és a fejlesztési vágyó amatőr a PRIMO működtető programját saját céljainak megvalósításában, adott feladatok megoldásában fel tudja használni.

Az otthoni mikroszámítógép alkalmazása mindenki számára nyitott. Beépítheti saját fejlesztésű készülékébe, felhasználhatja saját maga vagy más által előállított rendszerekben. Építhet hozzá kiegészítő elemeket, perifériákat, mérőműszereket, érzékelő és beavatkozó elemeket, működtetheti a legkülönbözőbb vezérlési és szabályozási folyamatokban. A gépre írt programokban felhasználhatja a működtető program rutinjait, az azok működését befolyásoló paraméterek értékeit saját céljainak megfelelően módosíthatja.

A szoftver leírás figyelmes olvasásával megismerheti a PRIMO rendszerprogramjának működését. Megvizsgálhatja hogy a mikroszámítógépet alkalmazhatja-e, s ha igen, milyen módon saját feladatainak megoldásában.

Reméljük, kérdéseire e füzet választ ad. Ha mégsem, keressen meg minket! Mint a berendezés létrehozói, műszaki és programozási kérdéseinek megoldásához készségesen rendelkezésére állunk. Ha együtt akar velünk működni, és ötleteire, javaslataira, műszaki megoldásaira, programjaira széleskörű érdeklődést lát, mutassa be azokat nekünk. Ezek gyártásbavitelét megszervezzük, terjesztését megoldjuk.

Legyen partnerünk!

COSY Műszaki Fejlesztő Leányvállalat

A PRIMO OTTHONI SZÁMÍTÓGÉP KÉT SZOLGALATI TALÁLMAN YON ALAPUL. EZEK TULAJDONOSA A MAGYAR TUDOMÁNYOS AKADEMIA SZÁMÍTASTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE. A TELJES BERENDEZÉST A MICROKEY KUTATÁSI FEJLESZTÉSI TERMELÉSI TÁRSASÁG FEJLESZTETTE KI. A TÁRSASÁG TAGJAI:

- MAGYAR TUDOMÁNYOS AKADEMIA SZÁMÍTASTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZET
- ELEKTRÓMODUL ELEKTRONIKAI ALKATRÉSZ KERESKEDELMI VÁLLALAT
- "ÚJ ÉLET" MEZŐGAZDASÁGI TERMELŐSZÖVETKEZET BARISAP

A MIKROSZÁMÍTÓGÉP MŰSZAKI, MENEDZSELÉSI KÉRDÉSEIVEL AZ MTA SZTAKI CSOP MŰSZAKI FEJLESZTŐ LEÁNYVÁLLALAT FOGLALKOZIK.

E LEÍRÁS A PRIMO '84.1 VERZIÓJÁT ISMERTETI. A FEJLESZTŐK ÉS A GYÁRTÓ FENNTARTJA AZON JOGAT, HOGY A BERENDEZÉST ÉS ANNAK MŰKÖDTETŐ PROGRAMJÁT TOVÁBBFEJLESSZE, ÚJ LEHETŐSÉGEKET TEREMTSÉN, A MEGLELVŐKET MÓDOSÍTSA. A LEÍRÁSBAN TALÁLHATÓ PROGRAMOK A PRIMO SZÁMÍTÓGÉPET VEZÉRLIK. A FEJLESZTŐK NEM SZÁVATOLJAK, HOGY AZ ALKALMAZOTT MEGOLDÁSOK EREDETI KÖRNYEZETÜKBŐL KIEMELVE HELYESEN MŰKÖDNEK. AZ EBBŐL EREDE VALAMENNYI KÁR A FELHASZNÁLÓT TERHELI.

A KIADÓ ÉS A SZERZŐK ELŐZETES ENGEDÉLYE NÉLKÜL TILOS A LEÍRÁST, VAGY ANNAK RÉSZÉIT SOKSZOROSÍTANI, VISSZAKERESŐ RENDSZERBEN TÁROLNI, VAGY BÁRMILYEN MÁS MÓDON SOKSZOROSÍTÁS CÉLJÁBÓL LEMÁSOLNI.

BEVEZETÉS

E könyv megírásakor célunk az volt, hogy megismertessük az Olvasóval a PRIMO otthoni mikroszámítógép ROM-ban tárolt rendszerprogramjának működését. Nem kívántunk tankönyvet írni, még akkor sem, ha néhány kérdés kapcsán alapismeretek magyarázatát is megtalálhatják a leírásban. Ennek egyedüli célja, hogy az alkalmazott kifejezések általunk használt értelmezését megmagyarázzuk. Törekedtünk viszont arra, hogy mindazon ismeretek, amelyekre a PRIMO használata során szükségük lehet megtalálhatók legyenek ebben a leírásban, ne kényszerüljenek azt több helyről összegyűjteni. (Természetesen figyelembe vettük, hogy a "PRIMO füzetek" sorozatban már megjelent a PRIMO hardver leírás. Így a számítógép áramköri részleteinek működésére nem térünk ki, ahol szükséges volt, az említett könyvre hivatkoztunk.)

A PRIMO szoftver leírás három részre tagolódik:

1. Rész PRIMO BASIC nyelv leírás

Ebben a részben a PRIMO '84.1 verziója által ismert BASIC programozási nyelv leírása található. A leírásban ismertetjük a BASIC nyelv alapvető jellemzőit, működési módjait, a programsorok formátumát, a konstansok, változók és kifejezések alkalmazásának szabályait. Megadjuk a PRIMO által értelmezett valamennyi parancs, utasítás és függvény alkalmazásának formai szabályait, azok értelmezését, a helytelen alkalmazás esetén megjelenő hibajelzéseket. Ahol az célszerűnek bizonyult, hosszadalmas magyarázat helyett, egyszerű példák segítségével mutatjuk be a BASIC nyelv egyes elemeinek használatát.

2. Rész A PRIMO számítógép működése

A 2. Részben a BASIC Interpreter, az Aritmetika és a Periféria kezelő programok működését ismertetjük. Leírjuk a BASIC program értelmezésének és végrehajtásának főbb lépéseit, a program memórián belüli tárolásának formátumát, az Interpreter működése során létrejövő táblázatok felépítését. Példák segítségével mutatjuk be, hogyan tárolja a PRIMO a különböző típusú numerikus változókat, a karaktersorozatokat. Megadjuk az alapvető aritmetikai műveleteket, függvényeket végrehajtó rutinok belépési címait, paramétereit. Ismertetjük a periféria kezelő programok működését, a felhasználásukhoz szükséges memória címek értékét, valamint értelmezését.

Függelék

A függelék a könyv terjedelmének csaknem felét teszi ki. Itt található a PRIMO BASIC hibajelzéseinek teljes listája, a karakter- és vezérlő kódok felsorolása, a billentyűzet gombjainak belső kódja. Ide került a Felhasználói karakterek definiálását, a hanggenerátor paraméterezését ismertető fejezet. Megtalálható a Függelékben a memória felosztása, a rendszer- és Interpreter változók címe, értelmezése, a kulcsszavak belső kódja is. Végezetül a PRIMO valamennyi periféria kezelő programjának megjegyzésekkel ellátott forrásnyelvi listáját is a Függelék tartalmazza.

Könyvünkben az Olvasó gondos munkánk ellenére is találhat hiányosságokat. Kérjük, észrevételeiket juttassák el hozzánk, megkönnyítve ezzel egyrészt a PRIMO otthoni mikroszámítógép fejlesztését, másrészt a mind pontosabb leírások készítését.

A SZERZŐK

1. RÉSZ

PRIMO BASIC nyelv leírás

1. Fejezet

A PRIMO BASIC alapvető jellemzői

1.1. Működési módok	I-1
1.2. Programsor formátum	I-2
1.2.1. Sorszámok	I-2
1.3. Karakterkészlet	I-2
1.4. Konstansok	I-3
1.4.1. Egyszeres és duplapontos konstansok	I-4
1.5. Változók	I-5
1.5.1. Változónevek és típusazonosító karakterek	I-5
1.5.2. Tömbváltozók	I-6
1.5.3. A változók tárolóigénye	I-6
1.6. Automatikus típuskonverzió	I-7
1.7. Kifejezések és műveleti jelek	I-8
1.7.1. Aritmetikai műveletek	I-8
1.7.1.1. Nullával osztás, túlcsordulás és hiányzó operandus.	I-9
1.7.2. Relációs műveletek	I-9
1.7.3. Logikai műveletek	I-9
1.7.4. Függvények	I-11
1.7.5. Karakteres műveletek	I-11
1.7.6. Műveletek végrehajtási sorrendje	I-12
1.8. Programszerkesztés	I-13
1.8.1. Program írás	I-13
1.8.2. Sorszerkesztés	I-14
1.9. Különleges billentyűk	I-16
1.10. Hibajelzések	I-17

2. Fejezet

PRIMO BASIC parancsok, utasítások és függvények

2.1. Parancsok	I-19
2.2. Utasítások	I-22
2.3. Függvények	I-46

----- FELJEGYZÉSEK -----

1. FEJEZET

A PRIMO BASIC alapvető jellemzői

Ebben a fejezetben röviden összefoglaljuk a **PRIMO BASIC** programozási nyelv alapvető jellemzőit. Ismertetjük a lehetséges működési módokat, a **BASIC** programsorok formátumát, a sorszámokat. Megadjuk a konstansok, változók és kifejezések alkalmazásának szabályait, a műveletek végrehajtási sorrendjét.

1.1. Működési módok

A **PRIMO** számítógép bekapcsolásakor a **PRIMO BASIC** azonnal bejelentkezik. A képernyő legfelső sorában megjelenik a következő felirat:

PRIMO BASIC SYSTEM '84.1

Ezután a képernyő következő üres sorának elején kiíródik az **Ok** felirat. Az **Ok** üzenet megjelenése jelzi, hogy a **PRIMO** képes a billentyűzetről begépelte parancsok fogadására és végrehajtására. Ezután a **PRIMO** parancs (direkt), vagy program (indirekt) üzemmódban használható.

Parancs módban a **PRIMO BASIC** parancsait és utasításait nem előzi meg sorszám. Az így - sorszám nélkül - begépelte sorban lévő utasítások végrehajtása a **RETURN** billentyű megérintése után azonnal megkezdődik. Az aritmetikai és logikai műveletek eredményei megjeleníthetők, vagy változóknak további felhasználásra tárolhatók, de maguk az utasítások végrehajtásuk után elvesznek. A direkt módot általában a **BASIC** program hibáinak keresésekor alkalmazzuk. Ha a **PRIMO**-t egyszerű kalkulátorként kívánjuk használni olyan számítások elvégzésére, amelyek nem igényelnek komplett **BASIC** programot, ismét csak a direkt módot alkalmazhatjuk.

Program módban a begépelte **BASIC** utasítássor elején sorszám áll. Az ilyen - sorszámmal kezdődő - sort a **RETURN** billentyű megérintése után a **PRIMO** a program memóriájában tárolja. A tárolt **BASIC** program a **RUN** parancs begépelésével indítható el.

----- PRIMO BASIC NYELV LEÍRÁS -----

1.2. Programsor formátum

A PRIMO BASIC programsorai a következő formátumúak (a szögletes zárójelben álló részek megadása opcionális):

nnnn BASIC utasítás [:BASIC utasítás]...(RETURN)

ahol nnnn a BASIC utasítássor decimális sorszáma. Egynél több BASIC utasítás is állhat egy programsorban, de ebben az esetben az egyes utasításokat egymástól a (:) karakterrel kell elválasztani.

Egy PRIMO BASIC programsor tehát mindig sorszámmal kezdődik és a sor végét a RETURN billentyű megnyomásával jelezzük. Egy programsor maximális hossza 210 karakter - 5 képernyő sor - lehet. (Valójában a programsor csak 209 karaktert tartalmazhat, a 210. karakter a programsor végét jelző kód számára van fenntartva.)

1.2.1. Sorszámok

Minden PRIMO BASIC programsor sorszámmal kezdődik. A sorszámok az egyes programsorok memóriabeli sorrendjét jelölik ki. A sorszámok ezen túlmenően a BASIC ugrásoknál és a BASIC program szerkesztésekor mint címkek is használatosak. A PRIMO BASIC programokban a sorszámoknak a 0...65529 tartományba eső decimális egész számoknak kell lenniük.

A LIST, LLIST, AUTO, DELETE és EDIT parancsokban - a későbbiek szerint - a programlista sorszámaira hivatkozunk. Ha a sorszám helyett egy pont (.) szerepel, az az aktuális sor sorszámát helyettesíti. (Az aktuális sor az utoljára futtatott, listázott, javított sor, ill. az a sor, amelyben a PRIMO BASIC az utolsó hibát találta.)

1.3. Karakterkészlet

A PRIMO BASIC karakterkészlete betűket, számokat, speciális jeleket és a Felhasználó által definiálható karaktereket tartalmaz.

A PRIMO BASIC betűnek tekinti az angol ABC kis és nagybetűit. (Nem használható betűként azonban a magyar ABC á, Á, é, É, í, Ó, ó, Ő, ő, ú, Ú, Ű, ű betűje. Ez utóbbi karakterek csak szövegkonstansokban szerepelhetnek.)

A PRIMO BASIC számjegykarakterei a 0...9 számjegyek.

Különleges karakterekként a PRIMO BASIC a következő jeleket értelmezi:

Karakter	Jelentése
	Betűköz
=	Egyenlőségjel vagy értékadás művelet
+	Összeadásjel vagy pozitív előjel
-	Kivonásjel vagy negatív előjel
*	Szorzásjel vagy csillag
/	Osztásjel vagy törtvonal
↑	Hatványjel vagy felfelé mutató nyíl
(	Nyitózárójel
)	Csukózárójel
%	Százalékjel vagy egész típusjel
#	Számjel vagy duplapontos típusjel
\$	Dollárjel vagy karakteres típusjel
!	Felkiáltójel vagy egyszerespontos típusjel
,	Vessző
.	Pont vagy tizedespont
"	Idézőjel
'	Aposztróf vagy :REM utasítás rövidítése
;	Pontosvessző
:	Kettőspont vagy BASIC utasításvégjel
&	"&S" jel vagy ampersand
?	Kérdőjel vagy PRINT utasítás rövidítése
<	Kisebbjel
>	Nagyobbjel

A Felhasználó által definiálható karakterek karakterkonstansokban a szabványos karakterekkel azonos módon használhatók.

1.4. Konstansok

A konstansok olyan értékek, melyek a PRIMO BASIC program végrehajtása során értéküket nem változtatják meg. A konstansokat két típusba sorolhatjuk: karakterkonstans és numerikus konstans.

A karakterkonstans - másnéven string - betűk, számok, speciális- és Felhasználó által definiált karakterek idézőjelek ("") közé zárt maximálisan 255 karakter hosszú sorozata.

Példák: "Ez egy karakter-konstans"
 "1985. III. 11."
 "Az aposztróf (') speciális karakter"

A numerikus konstans pozitív vagy negatív szám. A PRIMO BASIC numerikus konstansai nem tartalmazhatnak vesszőt, a tizedes vessző helyett tizedes pontot kell alkalmazni. A numerikus konstansoknak három formájuk van:

----- PRIMO BASIC NYELV LEÍRAS -----

Egész szám	-32768 és 32767 közé eső egész szám. Az egész szám sem törtrészt, sem tizedespontot nem tartalmazhat.
Fixpontos szám	Pozitív vagy negatív valós - tizedestörtet vagy tizedespontot tartalmazó - szám.
Lebegőpontos szám	Pozitív vagy negatív normál - másnéven exponenciális - alakban megadott szám. Egy lebegőpontos szám előjeles egész vagy fixpontos számból (mantissza), egy E betűből és az ezt követő előjeles egész számból (10-es alapú hatványkitevő) áll. A lebegőpontos szám kitevőjének lehetséges értéke: $-38 \leq x \leq 37$.

Példák: $187.938E-7 = 0.0000187938$
 $-.5E8 = -50000000$
 $-3.2E+4 = -32000$

(Dupla pontosságú lebegőpontos konstans alkalmazásakor a kitevő előtt álló E betű helyett D betűt kell írni.)

1.4.1. Egyszeres és duplapontos konstansok

Egy valós numerikus konstans a PRIMO BASIC-ben egyszeres- vagy duplapontos lehet. Az egyszeres pontos konstansok 6 számjegy pontossággal tárolódnak és 6 számjegyre pontosan írhatók ki, míg a duplapontos konstansok 16 számjegy pontossággal tárolódnak és 16 számjegyre pontosan írhatók ki. A PRIMO BASIC a számkonstansokat alapértelmezésben egyszeres pontosaknak tekinti.

Egyszeres pontos mindazon numerikus konstans, melyre igaz az alábbi megállapítások valamelyike:

- A számkonstans kitevő és/vagy típusazonosító nélkül megadott egész vagy valós szám.
- Normál alakban megadva a kitevő előtt E áll.
- A számkonstans végén egy felkiáltójel (!) áll.

Példák: 34578
24.2356
+242.35E23
345!

Duplapontos mindazon numerikus konstans, melyre igaz az alábbi megállapítások valamelyike:

- Normál alakban megadva a kitevő előtt D áll.
- A számkonstans végén egy kettőskereszt (#) áll.

Példák: 1643768531#
 -6.04567D+12
 4562.12367234#

1.5. Változók

Változóknak nevezzük a BASIC program azon szimbólumait, amik olyan mennyiségeket jelölnek, amelyek értéküket megváltoztathatják. A változókat változónevekkel jelöljük. A változó értéket kaphat a programozótól (lsd. direkt üzemmód), vagy egy művelet eredményeként a BASIC programban. Mielőtt egy változó értéket kapna, számértéke nulla.

1.5.1. Változónevek és típusazonosító karakterek

A PRIMO BASIC változónevei tetszőleges hosszúságúak lehetnek, de a rendszer csak az első két karaktert veszi figyelembe. A változónevek az angol ABC kis- és nagybetűiből, valamint számjegyekarakterekből állhatnak, de az első karakternek kötelezően betűnek kell lennie. A változónév végén még egy speciális típusazonosító jel is állhat, melyek használatát alább ismertetjük.

A változónév nem lehet azonos egy PRIMO BASIC kulcsszóval és a változónév belsejében sem állhat kulcsszó. A PRIMO BASIC valamennyi parancs-, utasítás-, függvény- és operátorneve kulcsszó.

Egy változó numerikus vagy karakteres értéket képviselhet. A karakteres változónevek végén egy dollárjelnek (\$) kell állnia. A dollárjel egy típusazonosító karakter, amely azt jelzi, hogy az adott változó karakteres értéket képvisel.

A numerikus változók egész-, egyszerespontos- és duplapontos értékeket jelölhetnek. A különböző típusú változókhoz rendelt típusazonosító karakterek a következők:

Z	Egész típusú változó
!	Egyszerespontos változó
#	Duplapontos változó

Amennyiben egy változónév végén nincs típusazonosító karakter, úgy a változó egyszerespontos számértéket képvisel.

Lásd.
DEF utasítás

Példák helyes változónevekre:

ALFA!	Egyszerespontos valós értéket jelképez
Minimum#	Duplapontos valós értéket jelképez
ix	Egész értéket jelképez
SZ\$	Karakteres értéket jelképez
X	Alapértelmezés, egyszerespontos értéket jelképez

Példák helytelen változónevekre:

Ar	ékezetes karaktert tartalmaz
1B	Nem betűvel kezdődik
TONNA	Kulcsszót (TO) tartalmaz
X(	érvénytelen típusazonosító karakterrel végződik

A PRIMO BASIC lehetővé teszi a változók típusazonosításának egy másik módját is. A DEFINT, DEFSNG, DEFDBL és DEFSTR utasítások segítségével egy-egy változó, vagy változók egész csoportjának típusa határozható meg. Az említett utasítások részletes leírása a 2.2. fejezetben található.

1.5.2. Tömbváltozók

Egy tömb a változó értékek olyan csoportja, amelyekre azonos névvel hivatkozhatunk. A tömbön belüli valamennyi érték kijelölhető úgy, hogy a változónév után zárójelek között megadunk egy egész értéket, vagy egy egész értéket szolgáltató kifejezést (az úgynevezett indexet). Az egész érték a tömb egy elemi változóját választja ki. Egy tömbváltozónév után zárójelek között, egymástól vesszővel elválasztva több index is megadható. Ilyenkor többdimenziós tömbökről beszélünk. Az index legkisebb értéke 0.

Példák: T(5) a T egydimenziós tömbváltozó hatodik elemére utaló változónév. (A tömb indexszámozása a 0 értéktől kezdődik, ezért jelöli 5 a hatodik elemet.)
V(3,8) a V kétdimenziós tömbváltozó negyedik sorának kilencedik elemére utaló változónév.

Az indexek - és így a dimenziók - maximális száma 255 lehet. A tömbváltozók elemeinek számát a PRIMO memóriájának mérete korlátozza.

1.5.3. A változók tárolóigénye

Az alábbi táblázatból kiolvasható, hogy a különféle típusú változók tárolásához hány byte felhasználása szükséges. A megadott számok csak a változó által képviselt érték tárolásához szükséges tárolóterület méretét mutatják, a változó nevének és egyéb paramétereinek megőrzésére felhasznált memória méretét nem.

Típus	Tárolóterület (byte)
Egész	2
Egyszeres pontos	4
Dupla pontos	8
Karakteres	3 + karaktersorozat aktuális hossza

1.6. Automatikus típuskonverzió

Ha szükséges, a PRIMO BASIC a numerikus konstansokat és a változók aktuális értékeit automatikusan átalakítja egy másik típusra a következő szabályok figyelembevételével:

1. Ha egy numerikus konstans értékét egy numerikus változóhoz rendeljük (értékkadás) és a változó típusa eltér a konstans típusától, akkor a konstans értéke a változó típusa által meghatározott formában kerül tárolásra. (Ha egy karakteres változóhoz numerikus értéket, vagy egy numerikus változóhoz karakteres értéket kísérelünk meg rendelni, úgy TM - Type Mismatch, Típuskeveredés - hibajelzés keletkezik.)

Példa: 100 xz=76.548
110 PRINT xz
RUN
76

2. Az aritmetikai kifejezések kiértékelése során valamennyi aritmetikai és összehasonlító művelet operandusa azonos pontosságúra, mégpedig a kifejezésben szereplő legpontosabb ábrázolású érték típusára konvertálódik. A kifejezés végső eredménye is ezen legpontosabb számaábrázolási típusnak megfelelően keletkezik.

Példák: 40 D=6/7z
50 PRINT D
RUN
.857143

A művelet egyszerespontos való adatokon hajtódik végre és az eredmény is egyszerespontos való számként keletkezik.

70 X#=6z/7!
80 PRINT X#
RUN
.8571428656578064

A művelet egyszerespontos való adatokon hajtódik végre, de az eredmény duplapontos való szám lesz.

3. A logikai műveletek operandusait egész értékké konvertálják és a művelet eredményét is egész szám formájában adják vissza. Az operandusoknak a -32768...32767 intervallumba kell esniük, ellenkező esetben OV - Overflow, Túlcsordulás - hibajelzés keletkezik.

4. Amikor egy lebegőpontos érték egész értékké konvertálódik, a lebegőpontos szám törtrésze elvész.

Példa: 10 Iz=746.921
20 PRINT Iz
RUN
746

A konstans egyszerespontos való, de az eredmény egész lesz.

1 New
0 1892
de nem
dupla pontos
x# = 6#/7 80
x# = CDBL(6#)/7!
↑
80

----- PRIMO BASIC NYELV LEÍRAS -----

5. Ha egy duplapontos változóhoz egyszeresponthas értéket rendelünk, a keletkező duplapontos értéknek csak az első hat számjegye képvisel valódi értéket. Ennek az az oka, hogy az átkonvertált egyszeresponthas érték csak hat értékes jegyet tartalmazott.

Példa: 100 A=1/3
 110 B#=A
 120 PRINT A,B#
 RUN
 .333333 .3333333432674408

1.7. Kifejezések és műveleti jelek

Kifejezésnek nevezzük a karakteres vagy numerikus konstansok és változók műveleti jelekkel kombinált olyan sorozatát, amely a kifejezés kiértékelése után eredményül egyetlen karaktersorozatot vagy számértéket ad.

A műveleti jelek az operandusokon végrehajtandó matematikai vagy logikai műveleteket jelölnék ki. A PRIMO BASIC műveleti jelei a következő öt csoportba sorolhatók:

1. Aritmetikai műveletek
2. Relációs (összehasonlító) műveletek
3. Logikai műveletek
4. Függvények
5. Karakteres műveletek

A felsorolt csoportokat a következőkben ismertetjük.

1.7.1. Aritmetikai műveletek

Az aritmetikai műveletek végrehajtási sorrendje - precedenciája - csökkenő sorrendben a következő:

Műveleti jel	Művelet	Példa
↑	Hatványozás	X↑Y
-	Negálás (előjelváltás)	-X
*,/	Szorzás, osztás	X*Y X/Y
+, -	Összeadás, kivonás	X+Y X-Y

Ha a műveletek előbb felsorolt végrehajtási sorrendjét meg kívánjuk változtatni, úgy zárójeleket kell alkalmazni. A zárójelben álló műveletek elsőként hajtódnak végre. A zárójelen belül álló műveletek végrehajtási sorrendje azonos a táblázatban leírtakkal.

1.7.1.1. Nullával osztás, túlcsordulás és hiányzó operandus

Ha a kifejezés kiértékelése során nulla értékkel történő osztás történik, /0 - Nullával osztás - hibajelzés keletkezik és a program további végrehajtása megszakad.

Ha a kifejezés kiértékelése során túlcsordulás következik be, OV - Overflow, Túlcsordulás - hibajelzés keletkezik és a program további végrehajtása megszakad. (Az alkalmazható legnagyobb szám a $\pm 1.70141E+38$).

Ha egy kifejezés végén műveleti jel áll, MO - Missing Operand, Hiányzó operandus - hibajelzés után a program további végrehajtása megszakad.

1.7.2. Relációs műveletek

A relációs műveletek segítségével két értéket hasonlíthatunk össze. A művelet eredménye egyrészt "igaz" (-1), másrészt "hamis" (0) lehet. Ez az eredmény egy programon belüli elágazó utasításban (ld. IF utasítás) használható fel.

A relációs műveletek a következők:

Műveleti jel	Vizsgált reláció	Példa
=	Egyenlő	X=Y
(>)	Egyenlőtlen	X(>)Y
(<)	Kisebb	X(<)Y
(>)	Nagyobb	X(>)Y
(<=)	Kisebb vagy egyenlő	X(<=)Y
(>=)	Nagyobb vagy egyenlő	X(>=)Y

(Az egyenlőségjel egyben az értékadás művelet jele is.)

Amennyiben az aritmetikai és relációs műveleteket egyetlen kifejezésben kombináljuk, minden esetben először az aritmetikai műveletek végrehajtása történik meg. Például az

$$X+Y<(T-1)/Z$$

kifejezés értéke "igaz", ha $X+Y$ kisebb, mint a $T-1$ osztva Z -vel.

1.7.3. Logikai műveletek

A logikai műveletek segítségével többszörös relációkat, bitműveleteket - másnéven Bool műveletek - hajthatunk végre. A logikai műveletek bitenkénti eredményt szolgáltatnak, amely vagy "igaz" (nem nulla) vagy "hamis" (nulla) lehet. Egy kifejezésben a logikai műveletek az aritmetikai és a relációs műveletek után

----- PRIMO BASIC NYELV LEÍRAS -----

hajtódnak végre. A logikai műveletek igazságtáblája a következő táblázatból olvasható ki. A műveletek végrehajtásuk csökkenő sorrendjében kerültek ábrázolásra.

NOT	X		NOT X	Logikai "NEM"
	0		1	
	1		0	
AND	X	Y	X AND Y	Logikai "ÉS"
	0	0	0	
	0	1	0	
	1	0	0	
	1	1	1	
OR	X	Y	X OR Y	Logikai "VAGY"
	0	0	0	
	0	1	1	
	1	0	1	
	1	1	1	

Amint említettük a relációs műveletekkel a programban elágazásokat hozhatunk létre. A logikai műveletek segítségével két vagy több relációs műveleti eredmény kapcsolható össze, amely végül egyetlen "igaz" vagy "hamis" eredményt szolgáltatva felhasználható a feltételes elágazó utasításban.

Példák: 100 IF Q<500 OR ALFA=6 THEN 120
 200 IF I>=0 AND J<>50 THEN 330
 300 IF NOT S THEN 75

A logikai műveletek működésük során operandusaikat 16 bites, előjeles, 2-es komplement kódú egész értékekké alakítják. Az operandusoknak a -32768...32767 intervallumba eső értékeknek kell lenniük, máskülönben OV - Overflow, Túlcsordulás - hibajelzés keletkezik. A megadott logikai műveletek a 16 bites operandusokon bitenként - vagyis a 16 biten külön-külön - hajtódnak végre. A keletkező eredmény "igaz", ha a 16 bit bármelyikén részeredményként 1 keletkezett és "hamis", ha mind a 16 biten a részeredmény 0 volt.

Mint látható a logikai műveletek segítségével tetszés szerinti bitmintázat alakítható ki. Az AND művelet segítségével egy vagy több bitet "kimaszkolhatunk" egy bináris értékből, míg az OR művelettel két bináris értéket "összefésülhetünk". A NOT művelet segítségével egy bináris érték 1-es komplementjét állíthatjuk elő. Az alábbiakban az elmondottak illusztrálására néhány példát mutatunk be.

----- PRIMO BASIC NYELV LEÍRAS -----

39 AND 15 = 7	39 = 00000000000100111 15 = 0000000000001111 ----- 0000000000000111 = 7
-1 AND 7 = 7	-1 = 1111111111111111 7 = 0000000000000111 ----- 0000000000000111 = 7
10 OR 24 = 26	10 = 0000000000001010 24 = 0000000000011000 ----- 0000000000011010 = 26
30 OR 30 = 30	30 = 0000000000011110 30 = 0000000000011110 ----- 0000000000011110 = 30
NOT -2 = 1	-2 = 1111111111111110 ----- 0000000000000001 = 1
NOT 5 = -6	5 = 0000000000000101 ----- 1111111111111010 = -6

1.7.4. Függvények

A függvények segítségével a megadott argumentumon egy előre definiált műveletet hajthatunk végre. A PRIMO BASIC számos beépített függvénnyel rendelkezik. E műveletek argumentumait minden esetben a függvény nevét követő zárójelek közé kell helyezni. A PRIMO BASIC - amennyiben numerikus eredményt szolgáltató függvényértéket számít - egyszeres pontos valós számértéket ad vissza eredményül még abban az esetben is, ha a függvény argumentuma dupla pontos valós érték volt. A felhasználható függvények teljes listája a 2.3. fejezetben található.

1.7.5. Karakteres műveletek

Karakteres műveleteknek azokat a műveleteket nevezzük, amelyek numerikus érték helyett egy karaktersorozatot adnak eredményül. A PRIMO BASIC is rendelkezik néhány karakteres műveleti jellel, valamint számos karakteres függvénnyel.

Két karakterkonstans vagy két karakteres változó aktuális értéke egyetlen karaktersorozattá fűzhető össze a "+" jel segítségével (összefűzés=konzatenáció).

----- PRIMO BASIC NYELV LEÍRAS -----

Példa: 10 IS="Boldog" : JS=" év"
 20 PRINT IS+JS
 30 PRINT IS+" új"+JS
 RUN
 Boldog év
 Boldog új év

Karakteres értékek összefűzésénél ügyeljünk arra, hogy a létrehozandó karaktersorozat 255 karakternél ne legyen hosszabb, ellenkező esetben a PRIMO LS - Long String, Hosszú karaktersorozat - hibát jelez.

Akárcsak a numerikus értékek, a karaktersorozatok is összehasonlíthatók. A karakteres összehasonlítás műveleti jelei azonosak a már megismert numerikus összehasonlító műveletek jeleivel:

= () () (=) =

Karaktersorozat összehasonlítása az egyes karakterek ASCII kódjának összehasonlításával történik. Ha a két karaktersorozat valamennyi karaktere megegyezik, a két karaktersorozat egyenlő. Ha az ASCII kódok eltérők, a kisebb bináris értékű (bináris kódú) karaktert tartalmazó karaktersorozat lesz a kisebb. Ha az összehasonlítás során az egyik karaktersorozat véget ér, a rövidebb karaktersorozat lesz a kisebb. A karaktersorozat elején és végén álló betűköz karakterek is értékesek, résztvesznek az összehasonlításban.

Példák:	"AA"<"AB"	A karakterek ASCII kódjai
	"Egyenlők"="Egyenlők"	a Függelékben találhatók.
	"A&"<"A'"	
	"kg">"KG"	
	"Rövid"<"Rövidebb"	

A karaktersorozat összehasonlítással karaktersorozatokat névsorba rendezhetünk. Mivel azonban a magyar nyelv ékezetes karakterei az ASCII kódtáblázatban - és így a PRIMO kódkészletében is - nem a megszokott sorrendben állnak, így két karaktersorozat névsorbeli sorrendje csak abban az esetben dönthető el az összehasonlító műveletek segítségével, ha bennük ékezetes karakter nem szerepel.

Az összehasonlító műveletekben szereplő valamennyi karakterkonstanst idézőjelek közé kell zárni.

1.7.6. Műveletek végrehajtási sorrendje

A PRIMO BASIC műveleti jeleinek megismerése után az alábbi táblázat a kifejezés kiértékelés során alkalmazott teljes műveletvégrehajtási sorrendet mutatja be. A táblázatban a műveleti jelek csökkenő végrehajtási sorrendben állnak.

Végrehajtási sorrend	Művelet
1.	Zárójelben álló tag
2.	Függvények
3.	↑ (hatványozás)
4.	- (előjelváltás)
5.	*, /
6.	+, -
7.	Relációs műveletek
8.	NOT
9.	AND
10.	OR

Amennyiben egy kifejezésen belül azonos végrehajtási szinten lévő műveleti jelek állnak, a végrehajtás a kifejezésen belül balról jobbra történik.

1.8. Programszerkesztés

Programszerkesztésnek - editálásnak - nevezzük azt a tevékenységet, melynek során a PRIMO memóriájában őrzött BASIC programot megváltoztatjuk.

1.8.1. Program írás

A BASIC program szerkesztését megvalósító EDITOR - amely a PRIMO BASIC Interpreter programjának része - az Ok felirat megjelenését követően átveszi a PRIMO vezérlését és egészen a RUN parancs kiadásáig meg is tartja azt. Minden sort, amelyet begéptünk, az EDITOR dolgoz fel. Mindazon sorok, amelyek egy számmal kezdődnek, a BASIC program újonnan begépt sorainak minősülnek, és az EDITOR e sorokat sorszámuk növekvő sorrendjében tárolja a PRIMO memóriájában.

A begépt sorokat az EDITOR a következő eseteknek megfelelően dolgozza fel:

1. A sort beilleszti a BASIC programba. Ez akkor fordul elő, ha a sor elején szabályos sorszám áll (0...65529), és a sorszám után legalább egy, betűköztől különböző karakter áll.
2. Egy meglévő sort módosít. Ez akkor fordul elő, ha a BASIC programban van már ilyen sorszámú sor. Ekkor az EDITOR a régi sort törli a programból, és annak helyére az újonnan begépt sort helyezi.
3. Egy meglévő sort töröl. Ez akkor fordul elő, ha az újonnan begépt sor csak egy sorszámból áll és ilyen sorszámú sor már van a BASIC programban. Ekkor az EDITOR törli az adott sorszámú sort.

----- PRIMO BASIC NYELV LEÍRAS -----

Ha az egymást követő sorok begépelése során a PRIMO memóriája betelik, OM - Out of Memory, Betelt a memória - hibajelzés keletkezik, és a begépelte sor már nem kerül be a memóriába.

Egynél több utasítás is elhelyezhető egyetlen programsorba, de ebben az esetben az egyes utasításokat egymástól a kettőspont (:) karakterrel kell elválasztani.

Egy PRIMO BASIC programsor maximálisan 210 karakter (5 képernyősor) hosszú lehet. Ha a sor begépelése közben elérjük az ötödik képernyősor végét, a PRIMO további karaktereket már nem fogad el, helyette hangjelzést ad.

A gépelés közben tévesen leütött karaktereket a ← billentyű segítségével törölhetjük a sorból. A BRK, CLS, vagy a SHIFT és ← billentyűk egyidejű megnyomása az eddig begépelte teljes sor törlését eredményezi.

1.8.2. Sorszerkesztés

A BASIC program egy sorának tartalma az EDIT parancs segítségével módosítható. A parancs a következő formában adható meg:

EDIT sorszám

Ha az EDIT kulcsszó után sorszámot adunk meg, akkor az adott sorszámú sor, míg "." megadása esetén az aktuális sor szerkeszthető. Ha a megadott sorszámú sor nincs a BASIC programban, a PRIMO UL - Undefined Line, Nemdefiniált sor - hibajelzést generál. Ha létezik a megadott sorszámú sor, az EDITOR a képernyő következő sorától kezdve kiírja a szerkesztésre kiválasztott sort, majd a kurzort a következő üres képernyősor elejére állítja, és egy vezérlő billentyű megnyomására vár. Az alkalmazható vezérlő billentyűk a következők:

- Megérintésekor a kurzor egy pozícióval jobbra lép. Ennek hatására a régi sor aktuális karaktere átkerül az új sorba. Ha a → billentyűt folyamatosan nyomjuk, a leírt funkció ismétlődik. Ha a régi sornak már nincs további - az új sorba átmásolható - karaktere, a → billentyű ismételt megérintése már hatástalan.
- ← Hatására a kurzor egy pozícióval balra lép. Ennek következtében az új sor aktuális utolsó karaktere törlődik. Ha a billentyűt folyamatosan nyomjuk, a leírt funkció ismétlődik. Ha az új sorban már egyetlen karakter sincs, a ← billentyű ismételt megérintése már hatástalan.
- ↓ A szerkesztés újrakezdését jelzi. Hatására az eddigi módosítások elvesznek, ismét az eredeti állapotú régi

sor elejétől kezdhetjük az editálást. A könnyebb tájékozódás érdekében az EDITOR ismét megjeleníti az eredeti sort.

SHIFT/+ A változtatott sor szerkesztésének újrakezdését jelzi. Megérintése után az EDITOR kilistázza az eddigi változtatásokkal a régi sort, majd a képernyő következő sorának elejére állítja a kurzort. Ezután úgy dolgozhatunk tovább, mintha most kezdenénk a szerkesztést. (A SHIFT/+ funkcióval vissza tudunk lépni a sorban, ha észrevesszük, hogy elfelejtettünk egy szükséges módosítást.)

SHIFT/+ A szerkesztés végét jelzi. Hatására a régi sor még hátralévő karakterei átkerülnek az új sorba, majd a szerkesztés befejeződik.

RETURN A szerkesztés végét jelzi. Ha a szerkesztés megkezdésekor elsőként a RETURN billentyűt érintjük meg, a szerkesztés alatt álló sor módosítása nélkül befejeződik az editálás. Ha már előreléptünk a szerkesztett sorban, a RETURN megérintése az aktuális pontban lezárja az új sort, a régi sor hátralévő része elvész, majd befejeződik a szerkesztés.

BRK Kilépés a szerkesztésből az eddigi módosítások figyelembevétele nélkül. Megérintése után – anélkül, hogy a szerkesztett sor tartalma megváltozna – a PRIMO befejezi a szerkesztést, és újabb parancs megadására vár.

Ha a szerkesztés alatt nem vezérlőbillentyűt, hanem egy karakterbillentyűt nyomunk meg, akkor a szerkesztés alatt álló sor a megérintett billentyű által jelképezett karakterrel bővül. Így a régi sort újabb karakterek beszúrásával megváltoztathatjuk. Ha a sor bővítése közben elérjük a 210 karakteres sorhosszt, a PRIMO további karaktereket már nem fogad el, helyette hangjelzést ad.

Amennyiben a régi sor sorszámát a szerkesztés alatt megváltoztatjuk, az editálás lezárása után megmarad az eredeti sor, miközben – új sorszámmal – a memóriába kerül az új sor is.

Ha a PRIMO a BASIC program végrehajtása közben egy szintaktikai – formai – hibát észlel, megszakítja a program további futtatását és SN – Syntax Error, Formai hiba – hibajelzés után automatikusan megkezd a hibát okozó programsor szerkesztését. Ennek hatására – a hibajelzés után – megjelenik a hibás sor, és a kurzor a képernyő következő sorának elején villog, jelezve, hogy az EDITOR vezérlő billentyű megnyomására várakozik.

----- PRIMO BASIC NYELV LEÍRAS -----

Ha a PRIMO memóriájában tárolt teljes programot akarjuk törölni, a NEW parancsot alkalmazzuk. A NEW parancsot általában egy új program begépelése előtt célszerű használni.

1.9. Különleges billentyűk

A PRIMO billentyűzetén különleges funkciót megvalósító, ún. vezérlő billentyűk találhatók. Ezek közül néhány feladatáról már e fejezet megelőző részében is volt szó. Az alábbiakban részletesen ismertetjük valamennyi vezérlő billentyű funkcióját.

- RETURN** A begépelte sor végét jelzi. Hatására a kurzor a következő sor elejére lép.
- CLS** Képernyő törlés (CLS = Clear Screen). Megérintése után a képernyő tartalma törlődik, a kurzor a legelső sor legelső oszlopára - Home pozíció - áll. A CLS gomb megnyomására a PRIMO az addig begépelte, de RETURN-el még le nem zárt sort "elfelejti".
- UPPER** Betűváltó billentyű. Megérintése után az egyetlen karakterképet tartalmazó - betű - billentyűk az eddigi kisbetűk helyett nagybetűket jelképeznek, ill. az eddigi nagybetűk helyett ismét kisbetűket.
- SHIFT** Pillanatváltó billentyű. A SHIFT billentyű érintésével egyidőben megnyomott gomb vagy nagybetűt, vagy - ha a billentyű több jelet is tartalmaz - a felső karaktert jelképezi. A billentyű helyes használata: először megérintjük a SHIFT gombot, majd azt folyamatosan lenyomva tartva megérintjük a kiválasztott másik billentyűt.
- CTR** Kódmodosító billentyű (CTR = Control). A CTR billentyű folyamatos érintése után megnyomott egyfunkciós betű billentyű által szolgáltatott kód az eredeti értéknél 64-el kisebb lesz. (ld. PRIMO karakterkódok F-7 oldal) A CTR billentyű segítségével lehet vezérlőkódokat előállítani.
- ↓** Kódmodosító billentyű. A ↓ billentyű folyamatos érintése után megnyomott billentyű által szolgáltatott kód az eredeti értéknél 128-al nagyobb lesz. A ↓ billentyű segítségével lehet pl. a 128-as kódtól kezdődő Felhasználó által definiált karaktereket a billentyűzetről begépelni. (Figyelem! A ↓ billentyű a leírt kódmodosító hatást csak akkor fejt ki, ha a képernyőn látszik a kurzor. A BASIC program futása közben tehát csak akkor, amikor INPUT utasítás végrehajtása folyik.)
- BRK** Programfutást megszakító billentyű. (BRK = Break).

----- PRIMO BASIC NYELV LEÍRAS -----

Parancs módban megnyomva a **PRIMO** "elfelejti" az éppen begépelt sort és új sor megadására vár. Ha egy **BASIC** program végrehajtása folyik, a **BRK** billentyű megérintésére a program futása megszakad, és a **PRIMO** a következő sorban megjeleníti az alábbi üzenetet:

Break in nnnnn

Az üzenetben álló **nnnnn** azon **BASIC** programsor sorszáma, amelynek végrehajtása közben megnyomtuk a **BRK** billentyűt. A **BRK** megérintésével félbeszakított **BASIC** program a **CONT** paranccsal továbbindítható.

1.10. Hibajelzések

Ha a **BASIC** program futása közben a **PRIMO** valamilyen hibát észlel, megszakítja a programfutást, és hibajelzés után parancsmódba megy át. A hibajelzés alakja a következő:

?? Error in nnnnn

Az üzenetben a **??** helyén áll a hiba jellegére utaló, két karakterből álló kód, míg az **nnnnn** azon **BASIC** programsor sorszáma, amelyben a hiba előfordult. A **PRIMO BASIC** hibajelzéseinek teljes listája a Függelékben található.

----- FELJEGYZÉSEK -----

2. FEJEZET

PRIMO BASIC parancsok, utasítások és függvények

Ebben a fejezetben felsoroljuk a PRIMO BASIC valamennyi kulcsszavát és röviden ismertetjük ezek funkcióját. Az egyes kulcsszavaknál megadjuk felhasználásuk szintaktikai szabályait is. E szabályok ismertetésénél a következő jelöléseket alkalmazzuk:

- A NAGYBETŐKET, és - az alább ismertetett speciális karaktereken kívül megadott - írásjeleket pontosan a leírt módon kell a BASIC programban alkalmazni
- A hegyes zárójelek - (<,>) - között szereplő rész értelesszerűen a megfelelő szintaktikai elemekkel helyettesítendő
- A szögletes zárójelek - [,] - között álló szintaktikai elemek szerepeltetése opcionális
- A függőleges vonal - | - bal, ill. jobb oldalán álló szintaktikai elem közül vagy az egyik, vagy a másik alkalmazható
- A három pont - ... - előtt álló szintaktikai elemek többször is ismételhetők

*

2.1. Parancsok

Parancsoknak nevezzük a PRIMO BASIC azon kulcsszavait, melyek a BASIC program szerkesztésére, ill. vezérlésére szolgálnak. A parancsok programból történő végrehajtása után a program futása megszakad és a PRIMO parancs (direkt) módba megy át. Másképpen megfogalmazva: az indirekt módban alkalmazott parancsok megszakítják a BASIC program futását.

AUTO [(kezdősorszám)[,(növekmény)]]

- Automatikusan egy újabb sorszámot generál a RETURN billentyű minden megnyomása után.

Az AUTO parancs elsőnek a kezdősorszámként megadott sorszámot generálja, majd minden további sorszámot az előzőhöz képest a növekménnyel megnöveli. Mind a kezdősorszám, mind a növekmény értékének - ha nem adjuk meg - a PRIMO 10-et feltételez. Ha a kezdősorszám után egy vessző áll, de nem adtunk meg növekményt, a PRIMO egy korábbi AUTO parancsban utoljára megadott növekményt használ.

Az AUTO parancs egy csillagot ír az általa generált sorszám

----- PRIMO BASIC NYELV LEÍRAS -----

után, ha ilyen sorszámú sor már van a BASIC programban. A csillag megjelenése jelzi, hogy a régi sort a PRIMO a begépelendő új sorra fogja cserélni.

A sorszámok folyamatos generálása a BRK billentyű megnyomásával szakítható meg. Azt a sort, amelyben a BRK billentyűt megérintettük, a PRIMO már nem tárolja el. Az AUTO parancs megszakítása után a PRIMO visszatér parancs módba.

CONT

- A BRK billentyű megnyomása, vagy egy STOP/END utasítás végrehajtása után továbbindítja a BASIC programot.

Ha a program lefutása - vagy futásának megszakítása - után a BASIC programot módosítjuk, a programot a CONT parancssal továbbindítani nem lehet. Ha mégis megkíséreljük, CN - Can't Continue, Nem folytatható - hibajelzés jelenik meg.

DELETE [(sorszám)][-(sorszám)]

- Törli a BASIC programból a megadott sorszámok között elhelyezkedő sorokat.

Ha a programban a megadott sorszámú sor nem található, vagy a DELETE kulcsszó után egyetlen sorszámot sem adunk meg, FC - Illegal Function Call, Illegális funkcióhívás - hibajelzés keletkezik.

Ha csak egyetlen sorszámot adunk meg, a PRIMO törli a kiválasztott BASIC programsort. Ha két, kötőjellel elválasztott sorszámot szerepeltetünk, a PRIMO kitörli e két sort, és a köztük elhelyezkedő valamennyi programsort. Ha egy kötőjelet és egy azt követő sorszámot írunk a DELETE kulcsszó után, a program első sorától a megadott sorig terjedő sorok törlődnek a BASIC programból.

EDIT (sorszám)

- Szerkesztésre megnyitja a megadott sorszámú sort.

Az EDIT parancs működésének részletes leírása az 1.8.2. fejezetben található. *14 sz.*

LIST [(sorszám)][-(sorszám)]]

- Kilistázza a memóriában őrzött teljes BASIC programot, vagy annak kijelölt részét.

Ha az egyik sorszámot sem adjuk meg, a PRIMO a teljes prog-

ramot kilistázza. Ha csak az első sorszámot adjuk meg, csak a kiválasztott sor jelenik meg a képernyőn. Ha megadtuk az első sorszámot és a "-" jelet, de nem szerepel a második sorszám, a kiválasztott sor, és minden ennél nagyobb sorszámú sor megjelenik. Amennyiben a "-" jelet és a második sorszámot adtuk meg, a PRIMO a legkisebb sorszámú sortól a megadott sorig terjedő programrészt listázza ki. Ha mindkét sorszámot megadtuk, a BASIC program kiválasztott része fog megjelenni.

A listázás során a PRIMO megjeleníti a kiválasztott programrész első sorát, majd megvizsgálja, hogy nyomjuk-e valamelyik gombot a billentyűzeten. Amíg egy billentyűt meg nem érintünk, a következő sor listázása nem kezdődik el. Ha megnyomjuk valamelyik gombot, megjelenik a következő sor, és a PRIMO ismét a billentyűzetet vizsgálja...

A listázás a BRK billentyű megnyomásával megszakítható. Ekkor a PRIMO parancs módba megy át.

LLIST [(sorszám)[-[(sorszám)]]]

- A sornyomtatóra listázza a memóriában őrzött teljes BASIC programot, vagy annak kiválasztott részét.

Az LLIST parancs végrehajtásakor a PRIMO nem szakítja meg a listázást, ha nem nyomjuk a billentyűzet egyik gombját sem. Használatának további jellemzői a LIST parancsnál leírtakkal egyeznek.

NEW

- Kitörli a PRIMO memóriájában tárolt teljes BASIC programot és törli az összes BASIC változó értékét és definiált típusát.

RUN [(sorszám)]

- A PRIMO törli az összes BASIC változót, majd megkezd a memóriában tárolt BASIC program végrehajtását.

Ha megadjuk az opcionális sorszámot, a BASIC program végrehajtása a megadott sorszámú sortól kezdődik, míg ha a RUN parancs után nem áll sorszám, a legkisebb sorszámú BASIC sorral kezdődik a program végrehajtása. Ha a BASIC programban nincs a megadott sorszámmal programsor tárolva, UL - Undefined Line, Nemdefiniált sor - hibajelzés keletkezik.

indefiniált típusát (SNG-ve állítja)

----- PRIMO BASIC NYELV LEÍRAS -----

TRON TROFF

- Be- ill. kikapcsolja a BASIC program nyomkövetését.

A TRON parancs megadása után a PRIMO hegyes zárójelek közé zárva megjeleníti az éppen végrehajtott BASIC programsor sorszámát. A megjelenő sorszámokat megfigyelve eldönthetjük, hogy programunk várakozásunknak megfelelő úton halad-e, vagy a végrehajtás céljainktól különböző irányt vett-e.

A TROFF parancs kikapcsolja a nyomkövetést, vagyis alkalmazása után már nem jelennek meg a végrehajtott BASIC utasítások sorszámai. (A NEW parancs is kikapcsolja a nyomkövetést.)

A nyomkövetést vezérlő parancsokat a BASIC programba is elhelyezhetjük. Más parancsokkal ellentétben a TRON/TROFF végrehajtása nem szakítja meg a BASIC program futását. Így lehetővé válik a BASIC program egy kiválasztott részének nyomkövetése. A TRON és TROFF parancsok alkalmazásával a program belövésének ideje jelentősen csökkenthető.

2.2. Utasítások

Utasításoknak nevezzük a PRIMO BASIC azon kulcsszavait, melyek használata a BASIC programban előre meghatározott szabályok szerint végbemenő folyamatok lejátszódását váltják ki. Az alább felsorolt utasítások többsége parancsként - direkt módon - is alkalmazható.

Az utasításokat ABC sorrendbe csoportosítva soroljuk fel. Ez alól csak akkor teszünk kivételt, ha egy utasításhoz logikailag egy másik, az ABC sorrendben máshová tartozó utasítás kapcsolódik. (pl. IF-THEN-ELSE)

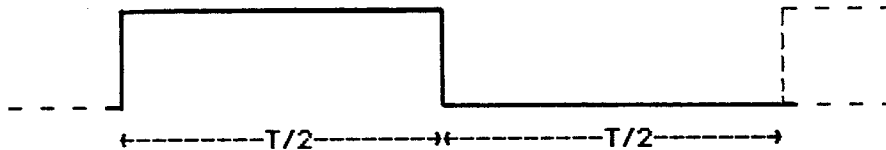
BEEP (pulzustényező), (pulzusszám)[; (pulzust.), (pulzussz.)]...

- Egy, a megadott paramétereknek megfelelő monofonikus hangot - vagy hangsorozatot - generál.

Az utasításban egy - vagy több - paraméterpár adható meg. A paraméterpár első tagja a generálandó hang frekvenciáját határozza meg. A paraméterpár második tagja a keltett hang periódusainak számát adja meg. Ez utóbbi lényegében a hang hossza. A BEEP utasítás paramétereinek a -32768...32767 tartományba kell esniük, ellenkező esetben a PRIMO OV - Overflow, Túlcserdülés - hibajelzést ad. Ha a megadott kife-

jezések aktuális értéke törtszám, a PRIMO az érték egész részével számol.

A megszólaló hang 50% kitöltési tényezőjű négyszögimpulzusok sorozata. Ha a jel periódusidejét T-nek nevezzük, a kialakuló jelforma a következő lesz:



Ahhoz, hogy egy kívánt frekvenciájú jelet elő tudjunk állítani, a következő kifejezést kell alkalmazni:

$$\langle \text{pulzustényező} \rangle = (T/2 - 35) / 8.4$$

Példa: 1000 Hz frekvenciájú, 10 sec időtartamú hang előállítás.

A periódusidő	= 1/1000 sec	→ 1000 mikrosec
$\langle \text{pulzustényező} \rangle$	= (500 - 35) / 8.4	→ 55
$\langle \text{pulzusszám} \rangle$	= 10 sec / 1000 mikrosec	→ 10000
az utasítás	= BEEP 55,10000	

CLEAR [(karakterterület méret)]

- A CLEAR utasítás törli a BASIC változók értékét, definiált típusát és - ha megadtuk az opcionális paramétert - beállítja a karaktersorozatokat (stringek) tárolására szolgáló memóriaterület méretét. (A leirtakon túl a CLEAR utasítás kikapcsolja a programozott hibakezelést, végrehajt egy RESTORE utasítást és törli a rendszer STACK-et. Ennek eredményeként a PRIMO megszakítja a FOR-NEXT ciklusok végrehajtását, valamint "elfelejti" a GOSUB utasítás végrehajtásakor eltárolt visszatérési címeket, így a RETURN utasítás hatására nem tud a főprogramba visszatérni. Ezért a CLEAR utasítást se ciklusban, se szubrutinban ne alkalmazzuk.)

A karakterterület méretének legalább akkorának kell lennie, mint a BASIC program végrehajtása során az összes karakteres változó által tárolt karaktersorozat együttes mérete, ellenkező esetben OS - Out of String Space, Betelt a karakterterület - hibajelzés keletkezik. Ekkor a hiba elhárítása végett - megfelelő argumentummal - adjunk meg egy CLEAR utasítást, majd indítsuk újra a programot. Ha a megadott kifejezés aktuális értéke törtszám, a PRIMO az érték egész részével számol.

----- PRIMO BASIC NYELV LEÍRÁS -----

A PRIMO bekapcsolásakor a karakterterület mérete automatikusan 50 byte lesz.

CLOSE

- Az utasítás lezárja a korábban OPEN vagy CREATE utasítással megnyitott adatállományt.

Ha a CLOSE utasítás végrehajtásakor nincs megnyitott adatállomány, a PRIMO FD - File Description, Állományleírás - hibajelzést generál. Ha az adatállományt a CLOSE utasítással nem zárjuk le, ez az adatok egy részének elvesztését eredményezi.

CLS

- Kitörli a képernyő tartalmát és a kurzort a bal felső sarokba állítja.

Ha a képernyő aktuális alapszíne fekete, a CLS hatására a teljes képernyő fekete lesz, míg fehér aktuális alapszín esetén a képernyőterület fehér lesz. A CLS utasítás minden esetben megszünteti a 6-os vezérlőkód - előtörölés - hatását.

CREATE (állománynév)

- Írásra megnyit egy adatállományt.

Az állomány megnyitása után a PRIMO a magnóra felírja az adatállomány nevét és típusát. Egy időben csak egyetlen állomány lehet megnyitva. A megnyitott állományba a PRINT# utasítással lehet adatokat írni.

Az állománynév megadása kötelező. A nevet karakteres konstanssal, karakteres változóval, vagy karakteres értéket szolgáltató kifejezéssel adhatjuk meg. Az állománynév maximum 16 karakter hosszú lehet, a kis-, ill. nagybetűket a PRIMO különbözőnek tekinti. Ha 16 karakternél hosszabb állománynevet adunk meg, FC - Illegal Function Call, Illegális funkcióhívás - hibajelzés keletkezik.

DATA (konstans)[,(konstans)]...

- A DATA utasítás a BASIC programban karakteres és/vagy numerikus konstansokat tárol, melyek a READ utasítással - vagy utasításokkal - olvashatók ki.

A DATA utasítások a programban tetszőleges helyen elhelyezhetők. Egy DATA utasítás egymástól vesszővel elválasztott, tetszőlegesen sok konstans tartalmazhat és a programban

tetszőlegesen sok DATA utasítás szerepelhet. A READ utasítás a DATA utasításokat sorszámuk növekvő sorrendjében olvassa ki, és az egyes DATA utasításokban álló konstansokat egyetlen összefüggő sorozatnak tekinti függetlenül attól, hogy hány konstans áll egy sorban, vagy hogy a programban hol helyezkedik el a következő DATA utasítás.

A DATA utasítást követő konstanslistában a numerikus konstansok egész, fixpontos, vagy lebegőpontos formában adhatók meg, de kifejezések nem szerepeltethetők. A karakteres konstansokat csak abban az esetben kell a DATA utasításban idézőjelek közé zárni, ha vesszőt, kettőspontot, vagy értékes vezető, ill. követő betűközöket tartalmaznak.

A READ utasításban szereplő változók típusának (numerikus vagy karakteres) meg kell egyeznie a DATA utasításban soronkövetkező konstans típusával, ellenkező esetben SN - Syntax Error, Formai hiba - hibajelzés keletkezik. A DATA utasítások kiolvasása a legkisebb sorszámú DATA utasítástól kezdődik. A RESTORE utasítás azonban ezt megváltoztathatja, megjelölve az először kiolvasandó DATA utasítást.

DEFDBL (kezdőbetű)[,(kezdőbetű)]...I(kezdőbetű)-(kezdőbetű)
 DEFINT (kezdőbetű)[,(kezdőbetű)]...I(kezdőbetű)-(kezdőbetű)
 DEFSGN (kezdőbetű)[,(kezdőbetű)]...I(kezdőbetű)-(kezdőbetű)
 DEFSTR (kezdőbetű)[,(kezdőbetű)]...I(kezdőbetű)-(kezdőbetű)

- Rendre duplapontos, egész, egyszeres pontos, ill. karakteres változóként definiál minden olyan változót, amely az utasításban megadott betűk egyikével kezdődik.

A kívánt kezdőbetűket kétféle formában is meghatározhatjuk:

- a kezdőbetűket egymástól vesszővel választjuk el
- kezdőbetű tartomány kezdő- és végkarakterét a "-" karakterrel kapcsoljuk össze.

Példák: DEFDBL A,B,K A,B,K kezdetű változók duplapontosak
 DEFINT I-N I,J...N kezdetű változók egészek
 DEFSTR S-U,Z S,T,U,Z kezdetű változók karakteresek

A DEFxxx utasítással típusazonosított változók BASIC programbeli használatakor nem szükséges a változónevek végén típusazonosító karaktert megadni. Ha mégis szerepeltetünk a változónév végén típusazonosító karaktert, az felülírja a DEFxxx utasításban kijelölt típust. A típusazonosító karakterek felsorolása az 1.5.1. fejezetben található.

DIM A(5,7,-)

DIM (indexelt változó)[,(indexelt változó)]...

- Meghatározza a programban használt tömbök dimenziószámát, és az egyes dimenziók maximális elemszámát, valamint helyet foglal a definiált tömb számára.

A dimenzionált tömb minden eleme kezdőértéket kap. Ez numerikus tömb esetén a 0, karakteres tömb esetén az üres string.

A PRIMO BASIC-ben egy tömb maximálisan 255 dimenziós lehet, a tömb összes elemének számát csak a memória mérete korlátozza.

Ha egy olyan indexelt változót alkalmazunk, amely még nem szerepelt egy DIM utasítás indexelt változó listájában, akkor a PRIMO ezen tömb minden dimenziójában a maximális elemszámot 10-nek tekinti. Abban az esetben, ha a tömb valamelyik indexe a későbbiekben meghaladja a 10-es értéket, BS - Bad Subscript, Hibás index - hibajelzés keletkezik. Az indexek minimális értéke minden esetben 0.

Ha a DIM utasításban olyan tömb méretét próbáljuk meghatározni, amely korábban - akár automatikusan, akár általunk - már dimenzionált, DD - Double Dimensioned, Tömb újradienzionálás - hibajelzés keletkezik.

Ha a PRIMO memóriájában a tömb elhelyezéséhez nincs már elegendő hely, OM - Out of Memory, Betelt a memória - hibajelzés jelenik meg.

END

- Megállítja a BASIC program futását.

Az END utasítás a BASIC programban tetszőleges helyen állhat, valamint egy programban több END utasítás is alkalmazható. Ellentétben a STOP utasítással, az END hatására nem jelenik meg a "Break in nnnnn" üzenet, csak az Ok felirat. A program végén az END utasítás szerepeltetése opcionális.

ERROR (kifejezés)

- A BASIC program futása közben a megadott kódú hibajelzést váltja ki.

A kifejezésnek 0-nál nagyobbnak és 256-nál kisebbnek kell lennie. Ha a kifejezés értéke megegyezik egy szabványos PRIMO BASIC hibakóddal, megjelenik a megfelelő hibajelzés. Amennyiben a megadott értékhez nincs szabványos hibajelzés

----- PRIMO BASIC NYELV LEÍRAS -----

rendelve, UE - User Error, Felhasználói hiba - jelzés keletkezik. Ha a megadott kifejezés aktuális értéke törtszám, a PRIMO az érték egész részével számol.

FOR (változó)=x TO y [STEP z]

.

.

NEXT [(változó)[,(változó)]...]

- Ciklusszervező utasításpár. Lehetővé teszi, hogy a FOR és NEXT utasítások között elhelyezkedő, tetszőlegesen sok BASIC utasítást a FOR utasítás paraméterei által meghatározott alkalommal a PRIMO ismételten végrehajtsa.

Megjegyzés: A FOR és NEXT utasításokban ciklusváltozóként egész, vagy egyszeres pontos skalár változók szerepelhetnek. Az x, y és z betűk numerikus értéket szolgáltatató kifejezéseket jelölnek.

A FOR kulcsszó után álló változó a ciklusváltozó. Az első kifejezés (x) a ciklusváltozó kezdeti értéke, míg a második kifejezés (y) a változó végértéke. A FOR utasítást követő utasítások a soronkövetkező NEXT utasításig végrehajtódnak, majd a ciklusváltozó értéke a STEP kulcsszót követő kifejezés (z) értékével megnövekszik. Ezután a PRIMO megvizsgálja, hogy a ciklusváltozó értéke meghaladja-e a végértéket (y). Ha még nem, a program végrehajtása a FOR utasítást követő résztől ismétlődik. Amennyiben a ciklusváltozó pillanatnyi értéke meghaladja a végértéket, a BASIC program a NEXT utasítás után álló első utasítástól folytatódik. Ha a FOR utasításban nem adtuk meg a lépésközt, azt a PRIMO 1-nek tekinti.

Ha a lépésköz értéke negatív, a ciklusváltozó értéke a lépésközzel csökken. A ciklusból történő kilépés feltétele ebben az esetben az, hogy a ciklusváltozó aktuális értéke a végértéknél kisebb legyen. 0 értékű lépésköz végtelen ciklust eredményez.

A ciklus magja egyszer abban az esetben is végrehajtásra kerül, ha a ciklusváltozó kezdőértéke nagyobb, mint a végérték.

A FOR...NEXT ciklusok egymásba ágyazhatók, vagyis egy ciklus belsejében - magjában - újabb FOR...NEXT ciklusok állhatnak. Ha ciklusokat egymásba ágyazunk, minden ciklusnak a másiktól eltérő nevű ciklusváltozóval kell rendelkeznie. Egy ciklus magjában elhelyezkedő ciklusnak még a külső ciklusmag vége előtt be kell fejeződnie. Az egymásba ágyazott ciklusok azonos végponttal is rendelkezhetnek, ebben az esetben a

----- PRIMO BASIC NYELV LEÍRAS -----

ciklusváltozók egyetlen NEXT utasításban is felsorolhatók. A ciklusok egymásba ágyazásának mélységét csak a rendelkezésre álló memória mérete korlátozza.

A ciklus magjából feltételes vezérlésátadó utasítással még a ciklus "lejárta" előtt ki lehet ugrani. Tilos azonban egy ciklus magjába a FOR utasítás megkerülésével belépni.

A NEXT utasítás után nem kötelező a ciklusváltozó megadása. Ebben az esetben a NEXT az utoljára megnyitott ciklust zárja le. Ha a PRIMO a BASIC programban egy FOR utasítás végrehajtása előtt észlel NEXT utasítást, NF - Next Without FOR, FOR nélküli NEXT - hibajelzést generál.

Ha ciklusváltozóként duplapontos valós változót adunk meg, TM - Type Mismatch, Tipuskeveredés - hibajelzés keletkezik. Ha a ciklusváltozó helyén indexelt változó áll, SN - Syntax Error, Formai hiba - hibajelzés jelenik meg.

GOSUB (sorszám)

-
-

RETURN

- Ugrás a megadott sorszámú BASIC sorban kezdődő szubrutinba, és visszatérés a szubrutinból.

A megadott sorszámnak egy szubrutin első sorának kell lennie. A szubrutin a BASIC program bármely pontjáról, tetszőlegesen sokszor meghívható. Egy szubrutin másik szubrutint is hívhat, ily módon a szubrutinok egymásba ágyazhatók. Az egymásba ágyazás mélységét csak a rendelkezésre álló memória mérete korlátozza.

A RETURN utasítás a szubrutinból való visszatérésre szolgál. A RETURN hatására a vezérlés az utoljára végrehajtott GOSUB utasítást követő BASIC utasításra kerül vissza. Egy szubrutin egynél több RETURN utasítást is tartalmazhat, ha a szubrutin logikai felépítése ezt szükségessé teszi. Egy szubrutin a BASIC program bármely részén elhelyezkedhet, de ügyeljünk arra, hogy a főprogram ne "csorogjon rá" a szubrutinra. Ezt a szubrutin első sorát megelőző BASIC sorba elhelyezett GOTO, END vagy STOP utasítással érhetjük el. Ha a PRIMO RETURN utasítást egy GOSUB utasítás végrehajtása előtt észlel, RG - RETURN without GOSUB, GOSUB nélküli RETURN - hibajelzés keletkezik. Ha a BASIC programban nem szerepel a GOSUB kulcsszó után megadott sorszámú sor, UL - Undefined Line, Nemdefiniált sor - hibajelzés keletkezik.

GOTO (sorszám)

- Feltétel nélküli vezérlésátadás a BASIC program megadott sorszámú sorára.

Ha a kiválasztott BASIC sor egy végrehajtható utasítást tartalmaz, a program futása ezen utasítástól folytatódik. Ha az adott sorban egy nem végrehajtható utasítás - DATA, REM - áll, a programfutás ezen utasítást követő első végrehajtható utasítástól folytatódik. Ha a BASIC programban nem szerepel a GOTO kulcsszó után megadott sorszámú sor, UL - Undefined Line, Nemdefiniált sor - hibajelzés keletkezik.

```
IF (kifejezés) THEN (utasítás)[: (utasítás)]...I (sorszám)
    [ELSE (utasítás)[: (utasítás)]...I (sorszám)]
IF (kifejezés) GOTO (sorszám)
    [ELSE (utasítás)[: (utasítás)]...I (sorszám)]
```

- Végrehajt egy döntést, és a BASIC programot a döntés eredményétől függően a program két különböző pontja közül az egyiket folytatja.

Ha az IF kulcsszót követő kifejezés értéke nem nulla - logikai IGEN -, a program a THEN-ág - a THEN kulcsszó után álló programrész - végrehajtásával folytatódik (ha GOTO kulcsszót szerepeltetünk, a megadott sorszámú sortól folytatódik a végrehajtás). A THEN kulcsszót vagy egy ugrási cím - sorszám -, vagy egy, ill. egymástól kettősponttal elválasztott több BASIC utasítás követheti. Amennyiben az IF kulcsszót követő kifejezés értéke nulla - logikai NEM -, az ELSE-ág - az ELSE kulcsszót követő rész - hajtódik végre. Ha az IF utasításban nem adtuk meg az opcionális ELSE-ágot, a következő BASIC sortól folytatódik a program végrehajtása.

Az IF...THEN...ELSE utasítások egymásba ágyazhatók, vagyis egy THEN-, ill. ELSE-ágban újabb IF...THEN...ELSE utasítások állhatnak. Az egymásba ágyazás mélységét csak a BASIC programsor hossza - 210 karakter - korlátozza. Ha az egymásbaágyazott IF...THEN...ELSE utasításokban a THEN- és ELSE-ágak száma eltérő, minden ELSE az őt megelőző, hozzá legközelebb eső IF utasításhoz kapcsolódik. Pl. a következő utasítás,

```
IF X=Y THEN IF Y=Z THEN PRINT "X=Z" ELSE PRINT "X<>Z"
```

nem írja ki az X<>Z üzenetet, ha X<>Y-al. Csak akkor jelenik meg az X<>Z, ha X=Y és Y<>Z.

Ha a THEN- vagy ELSE-ágot sorszám követi, és a BASIC programban nincs a megadott sorszámmal kezdődő sor, UL - Undefined Line, Nemdefiniált sor - hibajelzés keletkezik.

----- PRIMO BASIC NYELV LEÍRAS -----

INPUT ["<magyarázó szöveg>"];(<változó>[,<változó>])...

- A BASIC program futása alatt adatbevitelt hajt végre a billentyűzetről.

Amikor a PRIMO a BASIC program végrehajtása során egy INPUT utasítást észlel, felfüggeszti a programfutást, és a képernyőn egy "?" karakter megjelenítésével figyelmezteti a Felhasználót, hogy adatbevitelre vár. Ha az utasításban magyarázó szöveget is megadtunk, az a "?" előtt jelenik meg. Ezután a kívánt adatok a billentyűzeten begépelhetők. Az adatsor végét a RETURN billentyű megnyomásával jelezhetjük.

A billentyűzetről bevitt adatokat rendre az INPUT utasítást követő változólista elemei - mint aktuális értéket - veszik fel. A változólista elemeit egymástól vesszővel kell elválasztani.

A változólistában numerikus és/vagy karakteres változók - beleértve az indexelteteket is - szerepeltethetők. A változók típusának rendre meg kell egyeznie a begépeltek típusával. (A begépeltek karakteres adatokat csak abban az esetben kell idézőjelek közé zárni, ha vesszőt, kettőspontot, vagy értékes vezető, ill. követő betűközöket tartalmaznak.)

Ha a begépeltek adat típusa és a változólistában soronkövetkező változó típusa eltérő - numerikus helyett karakteres - a "?REDO" üzenet jelenik meg, majd a PRIMO az adatok ismételt begépelésére vár.

Példa: 100 input "x, y";x,y
110 print X+Y
run
x, y? 55,szöveg
?REDO
x, y? 55,12
67
Ok

\$-be ':' nem lehet
mert ott és követ-
kezőket eldobja egész
a sor végéig.
\$-t kezdő space-
ket nem veszi be! belül és
avegen már beveszi

Ha a billentyűzetről több adatot gépelünk be, mint ahány változó az INPUT utasítás után található, a PRIMO az "?Extra ignored" üzenet megjelenítése után - a felesleges adatokat figyelmen kívül hagyva - folytatja a BASIC program végrehajtását.

Amennyiben kevesebb adatot adunk meg, mint ahány változó az INPUT után áll, a PRIMO két kérdőjel megjelenítésével jelzi, hogy további adatok begépelésére vár.

Az INPUT utasítás végrehajtását a BRK billentyű megnyomása-
\$-nél a csak RETURN-re nem változik meg a régi

val félbe lehet szakítani, ekkor a PRIMO visszatér parancs módba. Ha ezután begépeljük a CONT parancsot, a program végrehajtása ismét a megszakított INPUT utasítástól folytatódik.

Ha INPUT utasítást direkt módban próbálunk megadni, ID - Illegal Direct, Direkt módban nem alkalmazható - hibajelzés jelenik meg.

INPUT# (változó)[,(változó)]...

- Az utasítás adatok beolvasását teszi lehetővé egy korábban OPEN utasítással megnyitott adatállományból.

Az INPUT# utasítás működése - azzal a különbséggel, hogy az adatok forrása nem a billentyűzet, hanem a magnó - lényegében azonos az INPUT utasítás működésével, de természetesen az utasítás végrehajtásakor nem jelenik meg a képernyőn a "?" karakter, és nem adható meg az INPUT# után magyarázó szöveg. Egyebekben ld. az INPUT utasítást.

Ha a programban még nem nyitottuk meg az adatállományt, a PRIMO FD - File Description, Állományleírás - hibajelzést ír a képernyőre.

Amennyiben az adatállományban már nincs annyi adatelem ahány változó a változó listában áll, a PRIMO FD - File Description, Állományleírás - hibát jelez.

[LET] (változó)=(kifejezés)

- A megadott változó felveszi az egyenlőséggel jobb oldalán álló kifejezés aktuális értékét.

Az értékadás műveletében nem kötelező a LET kulcsszó szerepeltetése, a PRIMO az "=" jel megjelenéséből is felismeri, hogy értékadást kell végrehajtania.

A műveletben a változó helyén tetszőleges típusú, egyszerű vagy indexelt változó állhat. Ha a változó típusa és a kifejezés által szolgáltatott érték típusa eltér - vagyis az egyik numerikus, míg a másik karakteres - a PRIMO TM - Type Mismatch, Típuskeveredés - hibajelzést generál. Amennyiben nincs típuskeveredés, de a numerikus változó típusa és a kifejezés típusa eltérő (egész, egyszeres pontos, duplapontos), az 1.6. fejezetben leírtak szerint a PRIMO automatikus típuskonverziót hajt végre.

----- PRIMO BASIC NYELV LEÍRAS -----

LOAD [(programnév)]

4295

- Magnóról betölt egy BASIC programot.

A LOAD utasítás hatására a PRIMO kitörli a memóriájában őrzött BASIC programot - NEW -, majd beolvas a magnóról egy BASIC programot. Ha a LOAD kulcsszó után megadjuk a betöltendő program nevét, a PRIMO azt a BASIC programot olvassa be, amelynek neve a magnón soronkövetkező programok közül elsőnek bizonyul azonosnak a megadott névvel. A két név abban az esetben azonos, ha:

- egyenlő hosszúak és a karakterek rendre azonosak
- a megadott név rövidebb, mint a magnón soronkövetkező program neve, de a hosszabb név első részének karakterei rendre azonosak a rövidebb név karaktereivel.

b) A programnevet karakteres konstanssal, karakteres változóval, vagy karakteres értéket szolgáltató kifejezéssel adhatjuk meg. A programnév maximálisan 16 karakter hosszú lehet, a kis-, ill. nagybetűket a PRIMO különbözőnek tekinti. Ha 16 karakternél hosszabb nevet adunk meg, FC - Illegal Function Call, Illegális funkcióhívás - hibajelzés keletkezik. Ha a LOAD utasításban nem adunk meg programnevet, a PRIMO a magnón soronkövetkező programot tölti be a memóriába.

Betöltéskor a képernyő legalsó sorában a

bb hh FOUND: (programnév)

üzenet jelenik meg. bb a beolvasott blokkok száma, hh a hibás blokkok száma. Ha az üzenetben a FOUND helyett SKIP látható, a PRIMO egy, a megadottól eltérő nevű programot talált, amelyet átlép, hogy megtalálja a keresett programot.

(A LOAD utasítás segítségével lehet a képernyőre tölteni a korábban SAVE SCREEN utasítással magnóra kimentett képernyőtartalmat is. A LOAD alkalmas a magnószalagon lévő gépi kódú - nem BASIC - programok betöltésére is. Ilyen formátumú programot azonban a PRIMO BASIC interpretere nem tud létrehozni. Erre a célra egyéb programok - pl. ASSEMBLER - alkalmazhatók.)

LPRINT [(kifejezések listája)]

LPRINT USING (karakteres kifejezés);(kifejezések listája)

- A felsorolt kifejezések értékét a nyomtatóra listázza.

Az utasítások - azzal a különbséggel, hogy az adatok nem a képernyőn, hanem a nyomtatón jelennek meg - azonosak a PRINT

és **PRINT USING** utasításokkal. Ez utóbbiak részletes leírása a fejezet későbbi részében található.

ON ERROR GOTO (sorszám)

- Lehetővé teszi a **BASIC** programban előforduló hibák programozott kezelését, és meghatározza a hibakezelő rutin kezdősorszámát.

A programozott hibakezelés bekapcsolása után bármely hiba - még a direkt módon észlelt szintaktikai hiba is - ugrást eredményez a hibakezelő rutinra. Ha az **ON ERROR GOTO** utasításban megadott sorszámú sor nincs a **BASIC** programban, **UL - Undefined Line, Nemdefiniált sor** - hibajelzés keletkezik.

Ha a korábban már bekapcsolt programozott hibakezelést meg kívánjuk szüntetni, ezt az **ON ERROR GOTO 0** utasítás megadásával tehetjük meg. (A **RUN** és **NEW** parancs, valamint a **CLEAR** utasítás is kikapcsolja a programozott hibakezelést.) Az ezt követően észlelt hibák már a **BASIC** program futásának megszakadását, és a hibára jellemző üzenet megjelenését eredményezik. Ha az **ON ERROR GOTO 0** utasítást egy hibakezelő rutinban adjuk meg, a **BASIC** program további végrehajtása megszakad, és megjelenik a hibarutinba ugrást kiváltó hibára jellemző üzenet.

Ha a **PRIMO** a hibakezelést végző rutinban észlel hibát, megjelenik a hibajelzés, és a program további végrehajtása megszakad, vagyis a hibakezelő rutinban elkövetett hibák programozottan nem kezelhetők le.

A programozott hibakezelést végrehajtó **BASIC** programrész logikai végét a **RESUME** utasítás jelzi (részletesen ld. e fejezet későbbi részében). Ha egy hibakezelő rutinban a **PRIMO** a **BASIC** program fizikai végét - nincs már az imént végrehajtott programsornál nagyobb sorszámú sor - észleli, **NR - No RESUME, Lezáratlan hibakezelés** - jelzést generál.

*Resume
törli az
ERR változót*

ON (kifejezés) GOTO (sorszám)[,(sorszám)]...

ON (kifejezés) GOSUB (sorszám)[,(sorszám)]...

- Ugrás a sorszámlistában megadott sorszámok valamelyike által kijelölt **BASIC** programsorra. (Többirányú elágazás.)

A kifejezés aktuális értéke határozza meg, hogy a sorszámlista hányadik elemét kell ugrási címként alkalmazni. Például, ha a kifejezés aktuális értéke négy, a sorszámlista negyedik eleme által meghatározott **BASIC** programsortól folytatódik a program végrehajtása. (Ha a kifejezés aktuális értéke törtszám, a **PRIMO** az érték egész részével számol.)

----- PRIMO BASIC NYELV LEÍRAS -----

Az **ON...GOSUB** utasításban a sorszámlista minden elemének egy szubrutin kezdősorszámát kell megjelölnie.

Ha a kifejezés aktuális értéke nulla, vagy nagyobb, mint a sorszámlista elemeinek száma – de kisebb, mint 256 –, a program végrehajtása az **ON...GOTO/GOSUB** utasítást követő első végrehajtható utasítástól folytatódik. Ha a kifejezés aktuális értéke nagyobb, mint 255, **FC – Illegal Function Call, Illegális funkcióhívás** – hibajelzés keletkezik.

OPEN [(állománynév)]

- Olvasásra megnyit egy adatállományt.

Az állomány megnyitása után a **PRIMO** a magnón megkeresi a megadott nevű állományt. Egy időben csak egyetlen állomány lehet megnyitva. A megnyitott állományból az **INPUT#** utasítással lehet adatokat olvasni.

Az állománynév megadása, valamint a betöltés alatt a képernyő legalsó sorában megjelenő üzenet értelmezése azonos a **LOAD** utasításnál leírtakkal.

OUT (port-cim),(adat)[,(port-cim),(adat)]...

- Kiküld egy adatbyte-ot a megcímzett output portra.

A port-cimnek és az adatnak a 0...255 intervallumba kell esnie. Amennyiben az értékek valamelyike meghaladja az előírt számértéket, **FC – Illegal Function Call, Illegális funkcióhívás** – hibajelzés keletkezik. Ha valamelyik érték törtszám, a **PRIMO** a megadott érték egész részével számol.

POKE (memóriacim),(adat)[,(adat)]...

- Kiír egy – vagy több – adatbyte-ot a megadott memóriacimre. Ha több adatot adunk meg, azok a memória egymást követő címekre kerülnek.

A memóriába kerülő adatoknak a 0...255 intervallumba kell esniük, mert máskülönben **FC – Illegal Function Call, Illegális funkcióhívás** – hibajelzés keletkezik. A memóriacim értékének a -32768...32767 intervallumban kell lennie, máskülönben **OV – Overflow, Túlcserdülés** – hibajelzés keletkezik. Amennyiben valamelyik kifejezés aktuális értéke törtszám, a **PRIMO** az érték egész részével számol.

Ha a memória 32767 feletti részébe kívánunk adatokat elhelyezni, a memóriacimból 65536-ot ki kell vonni.

Példa: a kívánt memóriacím=42678
a megadandó memóriacím=42678-65536 → -22858

PRINT [(kifejezések listája)]

- A megadott kifejezések aktuális értékét a képernyőre listázza.

Ha a PRINT kulcsszót nem követi a kifejezések listája, a képernyőn egy üres sor jelenik meg. Ha megadtunk kifejezéseket, azok aktuális értéke megjelenik a képernyőn. A listában numerikus és/vagy karakteres értéket szolgáltató kifejezések állhatnak. (Ha karakteres konstansokat alkalmazunk, azoknak idézőjelek között kell állniuk.)

A kifejezéslistában egymást követő elemek megjelenítésekor azok képernyőn való elhelyezkedése a kifejezéslistában alkalmazott központosítás függvénye. A PRIMO BASIC a képernyősorokat 16 karakter szélességű zónákra osztja. A kifejezéslistában egy vessző (,) azt eredményezi, hogy a következő kiírásra kerülő érték a soronkövetkező zóna elején jelenik meg. (Ha a képernyősoron belül már nincs újabb zóna, a PRIMO a kiírást a következő sorban folytatja. Ha betelt a képernyő, a PRIMO a képtartalom alsó 15 sorát egy sorral felfelé tolja - scroll -.) A pontosvessző (;) hatására a következő kiírásra kerülő érték közvetlenül az előzőleg kiírt érték után fog megjelenni. (A PRIMO egy numerikus értéket csak akkor ír - pontosvessző alkalmazása esetén - közvetlenül az előzőleg kiírt érték mögé, ha a megjelenítendő szám még elfér a képernyősorban. Ellenkező esetben az érték a következő sor elején fog megjelenni.)

Ha a kifejezéslistát vessző, vagy pontosvessző zárja, a következő PRINT utasítás a kiírást ugyanezen képernyősor megfelelő pozíciójában kezdi - ha ott még van megfelelő mennyiségű hely -. Amennyiben a kifejezéslista végén se vessző, se pontosvessző nincs, a kiíratás után a kurzor a következő képernyősor elejére áll. Ha a kiíratandó adatok nem férnek el egyetlen képernyősorban, a PRIMO a megjelenítést a következő képernyősorban folytatja.

A kiírt számok után minden esetben egy betűköz karakter áll. Pozitív szám elején - az előjel helyén - egy betűköz, negatív szám elején a negatív előjel jelenik meg.

A megjelenítési formátumot az eddig leírtakon túl a TAB(i) függvénnyel is vezérelhetjük. A TAB(i) függvény a kifejezéslista következő elemének kiíratási pozícióját jelöli ki. (Részletesen lásd. a Függvényeket ismertető 2.3. Fejezetben.)

----- PRIMO BASIC NYELV LEIRAS -----

A PRINT kulcsszó a BASIC program írásakor a ? karakterrel helyettesíthető. A program listázásakor a ? helyett már a PRINT kulcsszó fog megjelenni.

```
PRINT$(sorcim),(oszlopcim),[(kifejezések listája)]  
PRINT$(sorcim),(oszlopcim), USING (karakteres kifejezés);  
                                (kifejezések listája)
```

- A megadott kifejezések aktuális értéket - a képernyő kiválasztott pontjától kezdődően - kilistázza.

A PRINT\$ utasítás annyiban különbözik a PRINT utasítástól, hogy segítségével a megadott kifejezések a képernyő tetszőleges pontjától kezdődően jeleníthetők meg. A \$ karakter után álló - egymástól vesszővel elválasztott - két kifejezés aktuális értéke a kiírás kezdőpontjának sor- és oszlopcímét határozza meg. A sorcimnek a 0...15, míg az oszlopcímnek a 0...41 intervallumba kell esnie. Amennyiben az értékek a megadott intervallumon kívülre esnek, FC - Illegal Function Call, Illegális funkcióhívás - hibajelzés keletkezik. Ha a megadott kifejezések aktuális értéke törtszám, a PRIMO az érték egész részével számol.

A képernyő legfelső sora a 0-ás című, a legalsó sor a 15-ös. Egy soron belül a balszélső karakterpozíció a 0-ás, a jobbszélső a 41-es. A PRINT\$ utasítás szempontjából egy sorban még akkor is 42 karakterpozíció van, ha a PRIMO éppen duplaszélességű karakterek megjelenítését végzi. A kurzor pozicionálása ebben az esetben is a normál méretű karaktereknek megfelelően történik. (Ha a képernyő 15. sorának 41. pozíciójába írunk, a karakter megjelenítése után megtörténik a képernyőtartalom eltolása - scroll -.)

Ha függőleges írásra való áttérés után alkalmazzuk a PRINT\$ utasítást, a kurzor pozicionálása az új helyzetnek megfelelően módosul. Ilyenkor a képernyőn 21 sor, soronként 32 karakter jeleníthető meg. Ennek megfelelően a sorcim és oszlopcím értéke is 0...20, ill. 0...31 lehet. Függőleges írásnál a 0-ás című sor a képernyő bal szélén kezdődik, a 20-as sor a jobbszálen, egy soron belül a 0-ás című karakterpozíció a képernyő alján, a 31-es a képernyő tetején található.

```
PRINT USING (karakteres kifejezés);(kifejezések listája)
```

- Az előírt formátum szerint megjeleníteni a megadott kifejezések aktuális értékét.

Az utasításban a kifejezéslista értelmezése azonos a PRINT utasításnál leírtakkal, a karakteres kifejezés pedig a meg-

jelenítés formátumát meghatározó karaktereket tartalmazza.

A **PRINT USING** utasítás használatakor - ha karakteres értéket kívánunk megjeleníteni - a következő megjelenítést vezérlő karaktereket alkalmazhatjuk:

- "!"

Jelzi, hogy a kifejezések listájában soronkövetkező karakteres kifejezés által szolgáltatott karaktersorozatnak csak az első karakterét kell megjeleníteni.

Példa: A\$="Első karakter"
 Ok
 PRINT USING "!";A\$
 E
 Ok

- "%n db betűköz"

Jelzi, hogy a kifejezések listájában soronkövetkező karakteres kifejezés által szolgáltatott karaktersorozatnak csak az első n+2 db karakterét kell megjeleníteni.

Ha a két % jel között egyetlen betűköz karakter sem áll, a karaktersorozat első két karaktere fog megjelenni, ha egy betűköz van, akkor három karakter fog megjelenni, stb... Ha a karaktersorozat hosszabb, mint a vezérlőmező által meghatározott hossz, a túlnyúló karakterek nem fognak megjelenni. Ha viszont a vezérlőmező hosszabb, mint a karaktersorozat, akkor az utóbbi teljes hosszában megjelenik és utána még annyi betűköz, hogy összesen a vezérlőmező hossza által meghatározott számú karakter jelenjék meg.

Példa: A\$="darabolás"
 Ok
 PRINT USING "% %";A\$
 darab
 Ok

A **PRINT USING** utasítás használatakor - ha numerikus értéket kívánunk megjeleníteni - a következő megjelenítést vezérlő karaktereket alkalmazhatjuk:

- "@"

A kettőskeresztrel számjegypozíciót reprezentálunk. A megjelölt számjegypozíciókat a PRIMO minden esetben kijelzi. Ha a

----- PRIMO BASIC NYELV LETRAS -----

kinyomtatandó érték kevesebb számjegyet tartalmaz, mint amennyit a vezérlőmezőben megadtunk, a szám előtt annyi betűköz karakter fog megjelenni, hogy a szám kitöltse a rendelkezésére álló területet. Amennyiben a megjelenítésre váró számérték nem fér el a vezérlőmező által kijelölt területen - több számjegyet tartalmaz -, megjelenik a teljes szám, de előtte - figyelmeztetésként - egy % karakter is.

A # karakterek között bárhol állhat egy decimális pont. A pont a # jelek sorozatát két részre osztva meghatározza, hogy a megjelenő számérték hány törtjegyet, és hány egészrész jegyet tartalmazzon. Ha kijelöltünk egészrész jegyeket, akkor a tizedespont előtt egy "0" akkor is megjelenik, ha a számérték egyébként csak tört számjegyeket tartalmaz. Ha a megjelenítendő számérték több törtjegyet tartalmaz, mint amennyit a vezérlőmezőben kijelöltünk, a PRIMO az aktuális számértéket a megadott számú törtjegyre kerekíti.

Példa: PRINT USING "##.###";76.49,5.45678,.17
76.490 5.457 0.170
Ok

Ha a számérték megjelenítését vezérlő mezőben az egészrész, és a törtrész számjegyeinek száma meghaladja a 24-et, a PRIMO FC - Illegal Function Call, Illegális funkcióhívás - hibajelzést generál.

- "+"

A pluszjel feltüntetése a vezérlőmező elején, ill. végén azt eredményezi, hogy a megjelenítendő számérték előjele - pozitív előjel is - a szám elején, ill. végén fog megjelenni.

Példa: PRINT USING "+###.##";12.45,-12.45
+12.5 -12.5
Ok
PRINT USING "###.##+";12.45,-12.45
12.45+ 12.45-
Ok

- "-"

A kötőjel feltüntetése a vezérlőmező végén azt eredményezi, hogy a negatív számértékek megjelenítésekor a PRIMO az előjelet a szám végére - jobb szélére - írja.

Példa: PRINT USING "###.##-";12.45,-12.45
12.45 12.45-
Ok

- **"**"**

A vezérlőmező elején feltüntetett ****** azt eredményezi, hogy a megjelenítendő szám elején és végén esetlegesen előforduló betűköz karakterek helyett a **PRIMO *** karaktereket fog kiírni. A ****** továbbá két megjelenítendő számjegyet is jelképez.

Példa: PRINT USING "***#.##-";12.45,-12.45
*12.45**12.45-
Ok

- **"\$\$"**

Egy számspecifikációs vezérlőmező elején feltüntetett **\$\$** azt eredményezi, hogy a megjelenő szám előtt egy **\$** is kiíródik. A **\$\$** továbbá két megjelenítendő számjegyet is jelképez. Exponenciális kijelzés esetén nem alkalmazható.

Példa: PRINT USING "\$\$#.##";7.5,-12.38
\$7.50-\$12.38
Ok

- **"**\$"**

A ****\$** egy vezérlőmező elején az előzőekben ismertetett két szimbólum - ******, **\$\$** - hatását egyesíti. A szám előtti betűközök helyén ***** karakterek és egy **\$** jelenik meg. A ****\$** továbbá három megjelenítendő számjegyet is jelképez.

Példa: PRINT USING "**\$.##";54.128
*\$54.13
Ok

- **",,"**

A számjegyspecifikáció egészrész mezejének belsejében elhelyezett egy - vagy több - vessző azt eredményezi, hogy a számérték megjelenítésekor a **PRIMO** minden, a tizedespontról balra haladva elhelyezkedő harmadik számjegy elé egy vesszőt fog kiírni. A vessző továbbá egy megjelenítendő számjegypozíciót is jelképez.

Példa: PRINT USING "##,#####.##";1.2345678D+5
123,456.78
Ok

- **"++++"**

Egy számjegyspecifikációs mező végén feltüntetett **++++** jel-sorozat exponenciális kijelzést eredményez. A megjelenítésre

----- PRIMO BASIC NYELV LEÍRAS -----

váró számérték mantisszája a ffff előtt álló vezérlőmezőnek megfelelő formátumban fog megjelenni, a kitevő pedig E_{xxx}, vagy D_{xxx} alakban. Szükség esetén – a vezérlőmezőben kijelölt tízesdesponthnak megfelelően – változik a kitevő értéke. Ha az előjelet a megismert vezérlőkarakterek segítségével nem a kiírásra kerülő szám végén jelenítettük meg, az egészrész számára megadott számjegypozíciók közül a PRIMO egyet az előjel részére használ fel.

Példa: PRINT USING "##.#####";576.13
5.76E+02
Ok
PRINT USING "#.#####-";-75.55
7.56E+01-
Ok
PRINT USING "#.#####";534.1,-643.12
0.53E+03-.64E+03
Ok

- Ha a kiíratandó számjegy nem fér el a vezérlőmező által kijelölt területen, egy Z jel jelenik meg a szám előtt. Hasonlóan, ha a rendelkezésre álló terület kerekítés miatt lesz szűk, a kerekített számérték előtt megjelenik a Z jel.

Példa: PRINT USING "##.##";574.18
z574.18
Ok
PRINT USING "#.##";9.997
z10.00
Ok

Ha a PRINT USING utasítás megjelenítést vezérlő karakteres kifejezésében az eddig felsorolt karakterektől eltérő jelet alkalmazunk, vagy a vezérlőkaraktereket nem az ismertetett formában használjuk, akkor a PRIMO ezeket a jeleket karakteres konstansoknak tekinti és a számértékek, ill. karakter-sorozatok megjelenítésekor a vezérlő kifejezés tartalmának megfelelően jeleníti meg.

Példa: A\$="év:#### Hó: % %. =## nap"
Ok
PRINT USING A\$;1984,"március",31
év: 1984 Hó: már. =31 nap
Ok

A PRIMO a megjelenítést vezérlő karakteres kifejezés egyes vezérlő mezőit rendre a kifejezéslistában soronkövetkező kifejezések aktuális értékének megjelenítéséhez használja

fel. Ha a soronkövetkező vezérlőmező karakteres érték megjelenítésére szolgál, a kifejezéslista következő elemének karakteres értéket kell szolgáltatnia, ha viszont a vezérlőmező numerikus, a soronkövetkező kifejezésnek is számértéket kell adnia. Ha ez nem következik be, a PRIMO TM - Type Mismatch, Tipuskeveredés - hibajelzést ad.

Ha a vezérlő karakteres kifejezés kevesebb mezőt tartalmaz, mint a kifejezéslista elemeinek száma, akkor a vezérlő karaktersorozat "kimerülése" után a PRIMO a következő kifejezés megjelenítéséhez a vezérlő karaktersorozatot ismét az elejéről kezdi értelmezni. Így a PRINT USING utasítás segítségével több, azonos formában megjelenítendő számhoz csak egyetlen vezérlő mezőt kell megadni. Ha a vezérlő karakteres kifejezés több mezőt tartalmaz, mint a kifejezéslista elemeinek száma, a PRIMO a felesleges mezőket figyelmen kívül hagyja.

Ellentétben a PRINT utasítással, a PRINT USING utasításban a kifejezéslista elemeit elválasztó vessző, vagy pontosvessző karakter nem eredményez a megjelenítés során különbséget. A kifejezéslista végén alkalmazott vessző, vagy pontosvessző hatása azonban azonos azzal, amit a PRINT utasításnál leírtunk.

PRINT# <kifejezések listája>

PRINT# USING <karakteres kifejezés>;<kifejezések listája>

- A megadott kifejezések aktuális értékét egy korábban CREATE utasítással megnyitott adatállományba listázza.

Az utasítások - azzal a különbséggel, hogy az adatok nem a képernyőre, hanem az adatállományba kerülnek - azonosak a PRINT és PRINT USING utasításokkal. Ez utóbbiak részletes leírása a fejezet előző részében található.

Ha a programban még nem nyitottuk meg az adatállományt, a PRIMO FD - File Description, Állományleírás - hibajelzést generál.

RANDOM

- Véletlen kezdőértékről újraindítja a véletlenszámgenerátort.

A PRIMO a véletlenszámgenerátornak egy véletlen kezdőértéket ad, így a RANDOM utasítást követően generált véletlenszámok egy új véletlenszám-sorozat elemei lesznek. A RANDOM utasítás alkalmazásával elérhető, hogy a PRIMO az egymást követő bekapcsolások után mindig más és más véletlenszám-sorozatot generáljon.

----- PRIMO BASIC NYELV LEÍRAS -----

READ (változó)[,(változó)]...

- Beolvas egy adatot a DATA-listából, és azt aktuális értéként a soronkövetkező változóhoz rendeli.

A READ utasítás csak akkor alkalmazható, ha a BASIC programban DATA utasításokat is szerepeltetünk. A READ utasítás megkeresi a DATA-lista következő elemét, és ezt az értéket a READ utasítást követő változók közül a soronkövetkezőhöz rendeli. A READ utasítást numerikus és/vagy karakteres változók - beleértve az indexelt változókat is - követhetik. Ha a READ utasításban a soronkövetkező változó numerikus, akkor a DATA-lista aktuális elemének is numerikusnak kell lennie. Abban az esetben, ha ez nem következik be, SN - Syntax Error, Formai hiba - hibajelzés keletkezik. Ha a READ utasításban van még értéket nem kapott változó, de a DATA-listából több adat már nem olvasható ki, OD - Out of Data, Kimerült a DATA-lista - hibajelzés keletkezik.

Ha újra akarjuk olvasni a DATA-lista tartalmát, a RESTORE utasítást kell alkalmazni. A RESTORE részletes leírása a fejezet későbbi részében található.

REM [(megjegyzés)]

- A BASIC program megértését elősegítő megjegyzést helyez a programba.

A REM utasítást a PRIMO a BASIC program végrehajtása során figyelmen kívül hagyja, de a program listázásakor változatlan formában megjeleníti. Egy GOTO, vagy GOSUB utasításban ugrási címként megadhatjuk a REM utasítás sorszámát is. Ebben az esetben a BASIC program végrehajtása a REM utasítást követő programsor első utasításától folytatódik.

A REM kulcsszó a BASIC program írásakor a ' karakterrel helyettesíthető. Az ' alkalmazása esetén a megjegyzést a sorban előrébb álló BASIC utasítástól nem kell kettőspont karakterrel elválasztani. Ellentétben a PRINT kulcsszót helyettesítő ? karakterrel, az aposztrof a program listázásakor sem alakul át REM kulcsszóvá.

Megjegyzés végének a PRIMO csak a BASIC sor végét tekinti. A REM utasítás tehát csak egy BASIC programsor utolsó utasítása lehet. Ha a REM kulcsszót az ' karakterrel helyettesítjük, ügyeljünk arra, hogy ezt DATA utasítás sorában ne tegyük, mert megjegyzésünket a PRIMO a DATA-lista elemének fogja tekinteni.

RESET ((x-koordináta),(y-koordináta))

- Kikapcsolja a képernyőn a megadott koordinátájú pontot.

A PRIMO képernyője 256x192 pontból áll. Ezek bármelyike a RESET utasítás segítségével kikapcsolható.

Az x-koordináta 0...255, az y-koordináta 0...191 értéket vehet fel. A képernyő bal alsó sarkában található a 0,0 koordinátájú pont, a jobb felső sarokban a 255,191 koordinátájú.

Ha a képernyő aktuális alapszine fekete, a RESET utasítás hatására a megcímzett pont is fekete lesz, míg ha a képernyő alapszine fehér, a címzett pont is fehér lesz.

Ha a koordinátaként megadott kifejezések valamelyike az említett intervallumon túli értéket képvisel, FC - Illegal Function Call, Illegális funkcióhívás - hibajelzés keletkezik. Ha valamelyik kifejezés aktuális értéke törtszám, a PRIMO az érték egész részével számol.

RESTORE [(sorszám)]

- Lehetővé teszi, hogy a DATA-listát - vagy annak kijelölt részét - újraolvassuk.

A RESTORE utasítás végrehajtása után a következő READ utasítás a DATA-listát az első elemtől kezdve olvassa ki. Ha megadjuk az opcionális sorszámot, a következő READ utasítás a megadott sorszámú sorban - vagy az azt követő sorok valamelyikében - álló DATA utasítástól kezdve olvassa a DATA-listát. Ha olyan sorszámot adunk meg, amely nem szerepel a BASIC programban, UL - Undefined Line, Nemdefiniált sor - hibajelzés keletkezik.

RESUME

RESUME 0

RESUME NEXT

RESUME (sorszám)

- A programozott hibakezelés befelyezését jelzi.

A felsorolt négy forma közül bármelyik alkalmazható, de a különféle RESUME utasítások hatása más és más.

A RESUME vagy RESUME 0 utasítás hatására a programozott hibakezelés végeztével a BASIC program végrehajtása a hibát okozó utasítástól folytatódik.

----- PRIMO BASIC NYELV LEÍRAS -----

A **RESUME NEXT** utasítás végrehajtása után a **BASIC** program a hibát okozó utasítást követő utasítástól folytatódik.

Ha a **RESUME** utasítás után egy sorszámot adunk meg, a programozott hibakezelés végeztével a **BASIC** program végrehajtása a megadott sorszámú sortól folytatódik. Ha a programban nincs ilyen sorszámú sor, **UL - Undefined Line, Nemdefiniált sor** - hibajelzés keletkezik.

Ha a **PRIMO** a hibakezelő rutinon kívül egy **RESUME** utasítást észlel, **RW - Resume Without Error, Hiba nélküli RESUME** - hibajelzést generál.

A programozott hibakezelést az **ON ERROR GOTO** utasítással lehet aktivizálni. Ennek részletes leírása e fejezet előző részében található.

SAVE (programnév) 1229

- Magnóra kimenti a **PRIMO** memóriájában lévő **BASIC** programot.

Az utasítás hatására a **PRIMO** a magnóra felírja a megadott programnevet, majd 256 byte-os részekre - blokkokra - osztva a szalagon rögzíti a memóriában pillanatnyilag tárolt **BASIC** programot.

A programnév megadása kötelező. A nevet karakteres konstanssal, karakteres változóval, vagy karakteres értéket szolgáltató kifejezéssel adhatjuk meg. A programnév maximálisan 16 karakter hosszú lehet, a kis-, ill. nagybetűket a **PRIMO** különbözőnek tekinti. Ha 16 karakternél hosszabb programnevet adunk meg **FC - Illegal Function Call, Illegális funkcióhívás** - hibajelzés keletkezik.

SAVE SCREEN (állománynév)

- A képernyő pillanatnyi tartalmát magnóra menti.

A **SAVE SCREEN** utasítás hatására a **PRIMO** a magnóra felírja a megadott állománynévet, majd 256 byte-os részekre - blokkokra - osztva a szalagon rögzíti a képernyő pillanatnyi tartalmát tároló - 6 Kbyte méretű - memória bitmintázatát. (Az elmentett képernyőtartalom később egy **LOAD** utasítással visszatölthető. A **LOAD** utasítás a beolvasott állomány típusából állapítja meg, hogy a szalagról érkező adatokat a képernyő-memóriába kell töltenie.)

A programnév megadása kötelező. A név megadásának szabályai azonosak a **SAVE** utasításnál leírtakkal.

SET ((x-koordináta),(y-koordináta))

- Bekapcsolja a képernyőn a megadott koordinátájú pontot.

Ha a képernyő aktuális alapszíne fekete, a SET utasítás hatására a megcímzett pont fehér lesz, míg ha a képernyő alapszíne fehér, a címzett pont fekete lesz.

Az x-koordináta a 0...255, az y-koordináta a 0...191 értéket veheti fel. A képernyő bal alsó sarkában található a 0,0 koordinátájú pont, a jobb felső sarokban a 255,191 koordinátájú. Ha a megadott értékek valamelyike az említett intervallumon túli értéket képvisel, FC - Illegal Function Call, Illegális funkcióhívás - hibajelzés keletkezik. Ha valamelyik kifejezés aktuális értéke törtszám, a PRIMO az érték egész részével számol.

STOP

- Megállítja a BASIC program futását, és a PRIMO visszatér parancs módba.

A STOP utasítás a BASIC program tetszőleges pontján szerepeltethető. Ha a program logikája megkívánja, a programban több STOP utasítás is elhelyezhető. Amikor a PRIMO végrehajt egy STOP utasítást, a képernyőn a következő üzenet jelenik meg:

Break in nnnnn

ahol nnnnn annak a BASIC programsornak a sorszáma, amelyben a megállást kiváltó STOP utasítás szerepelt.

A STOP utasítással megállított BASIC program a CONT parancs begépelésével továbbindítható.

TEST [(programnév)]

- Ellenőrzi a magnón rögzített program formátumát.

A TEST utasítás hatására a PRIMO a magnón megkeresi a megadott nevű programot, majd ellenőrzi annak formátumát. Ha a rögzítés hibátlan, a képernyőn megjelenő - a LOAD utasításnál ismertetettel azonos - üzenetben látható hibaszámláló értéke 0.

Mivel az ellenőrzés nem a magnón őrzött információ és a PRIMO memóriájában tárolt program összehasonlításán - hanem a szalagformátum ellenőrzésén - alapul, ezért nem feltétlenül szükséges, hogy az ellenőrzendő program a memóriában legyen.

----- PRIMO BASIC NYELV LEÍRAS -----

Ha a TEST kulcsszó után megadjuk az ellenőrizendő program nevét, a PRIMO azt a programot ellenőrzi, amelynek neve a magnón soronkövetkező programok közül elsőnek bizonyul azonosnak a megadott névvel. A két név akkor azonos, ha:

- egyenlő hosszúak és a karakterek rendre azonosak
- a megadott név rövidebb, mint a magnón soronkövetkező program neve, de a hosszabb név első részének karakterei rendre azonosak a rövidebb név karaktereivel.

Ha a TEST utasításban nem adunk meg programnevet, a PRIMO a magnón soronkövetkező program formátumát ellenőrzi. Ha megadjuk a program nevét, akkor azt a LOAD utasításban leírtak szerint tehetjük meg.

2.3. Függvények

Függvényeknek nevezzük a PRIMO BASIC azon kulcsszavait, melyek kifejezésekben változók, vagy konstansok helyén alkalmazhatók, és megadott argumentumukon egy előre definiált funkciót végrehajtva, eredményül egy számértéket, vagy karaktersorozatot szolgáltatnak. A beépített függvények többsége a műszaki-matematikai gyakorlatban megszokott egyértékű numerikus függvény, de a PRIMO számos, karakteres értékeken manipuláló függvénnyel is rendelkezik. Ezen túlmenően néhány speciális függvény is alkalmazható.

A leírásban a függvények argumentumát az *i*, *j*, *x* és *x\$* szimbólumokkal jelöljük. Az egyes szimbólumoknak a következő jelentést tulajdonítjuk:

- *i* és *j* numerikus egész értéket szolgáltató kifejezést jelöl. Ha a kifejezés aktuális értéke nem esik a -32768...32767 intervallumba, OV - Overflow, Túlcsordulás - hibajelzés keletkezik. Ha a megadott kifejezés aktuális értéke törtszám, a PRIMO az érték egész részével számol.

(Néhány függvény esetén a kifejezés értékének még az előbb említett intervallumnál is szűkebb tartományba kell esnie. E tartományok túllépése esetén a PRIMO FC - Illegal Function Call, Illegális funkcióhívás - hibajelzést generál. A szűkebb intervallumok méretét az egyes függvényeknél adjuk meg.)

- Az *x* egész-, egyszeresponstos- vagy duplapontos számértéket szolgáltató kifejezést jelöl.
- Az *x\$* karakteres értéket adó kifejezést jelképez. A karakteres kifejezések alkalmazásánál ügyeljünk arra, hogy a létre-

hozandó karaktersorozat nem lehet hosszabb 255 karakternél, ellenkező esetben a PRIMO LS - Long String, Hosszú karakter-sorozat - hibát jelez. A karakterkonstansokat, ill. karakteres változókat tartalmazó, a PRIMO számára túlságosan bonyolult kifejezések előfordulása esetén ST - String Formula Too Complex, Túlságosan bonyolult karakteres kifejezés - hibajelzés keletkezik. Ilyenkor a kifejezést két, vagy több részre osztva kell a BASIC programban leírni.

Azt, hogy egy függvény eredményül numerikus, vagy karakteres értéket szolgáltat-e, a függvény nevének formátumából is eldönthetjük. Mindazon függvény neve, amely eredményül karaktersorozatot ad, a \$ karakterrel záródik, míg a numerikus értéket adó függvénynevek végén nincs típusazonosító karakter.

ABS(x)

- Az x kifejezés aktuális értékének abszolútértékét eredményezi.

ASC(x\$)

- Az x\$ karaktersorozat első karakterének ASCII kódját adja eredményül. Ha az x\$ üres string - a hossza nulla - FC - Illegal Function Call, Illegális funkcióhívás - hibajelzés keletkezik.

ATN(x)

- Eredményül x arkusz tangensét adja radiánban. Az eredmény a $-pi/2...pi/2$ tartományba esik. Az x kifejezés tetszőleges numerikus típusú lehet, de az ATN függvény mindig - még duplapontos argumentum esetén is - egyszerespontos értéket szolgáltat. Ennek az az oka, hogy a PRIMO a logaritmikus, exponenciális, trigonometrikus és inverz trigonometrikus függvényeket - az egyszerűbb és gyorsabb számítás érdekében - csak 7 decimális jegy pontosságra számítja ki, és így a duplapontos valós szám további 9 decimális jegye már nem képviselne valódi értéket.

CALL(i[,j])

- Felhasználói gépi kódú rutint indít. Az első operandus a gépi kódú programrész memóriacíme, a második operandus értéke - ha megadjuk - a gépi kódú rutinba érkezéskor a Z80 mikroprocesszor DE regiszterpárjában található.

A gépi kódú rutint a Felhasználónak kell a memóriában elhelyeznie, és gondoskodnia arról, hogy a rutint a BASIC Interpreter működése során ne írja felül. A CALL függvényben a

----- PRIMO BASIC NYELV LEIRAS -----

gépi kódú rutin memóriacímét a POKE utasításnál leírt módon kell megadni.

A gépi kódú rutinban a Z80 mikroprocesszor belső regiszterei - az IX és I regiszterek kivételével - szabadon használhatók, mentésükről a gépi rutin meghívása előtt a PRIMO gondoskodik. Az IX és I regiszterek értéke azonban még időszaki-
kban sem módosítható, mert ez a PRIMO helytelen működéséhez vezetne.

Mivel a CALL függvény, működésének befejezéseként eredményül egy számértéket szolgáltat. A CALL mindig egy egész számot ad eredményül, amely a Z80 mikroprocesszor HL regiszterpárjának a gépi rutinból való visszatérést közvetlenül megelőző értéke lesz. A leírtakból kitűnik, hogy a gépi kódú rutinnak egy paramétert adhatunk át, és eredményül is egy paramétert kaphatunk vissza. (Természetesen az átadott paraméter jelölheti egy paraméterlista elejét is, így a gépi rutin már több paramétert is átvehet a BASIC programtól. Teljesen hasonló módon egynél több paraméterérték vissza is adható a BASIC programnak.)

A gépi kódú rutinból a BASIC programba a vezérlést a Z80 mikroprocesszor RET utasításával adhatjuk vissza. Ögyeljünk arra, hogy a gépi szintű rutin a STACK-be ne helyezzen el több adatot, mint amennyit onnan kiolvas, vagy ne olvasson ki a behelyezett adatoknál többet, mert ez a PRIMO helytelen működéséhez vezethet. (A gépi szintű rutin számára a PRIMO 40 byte méretű STACK területet biztosít. Ennél több adat STACK-be helyezése a számítógép helytelen működését eredményezheti.)

CDBL(x)

- Az x kifejezés aktuális értékét duplapontos számértékké alakítja, és eredményül ezt a duplapontos értéket szolgáltatja.

CHR\$(i)

- Eredményül 1 karakter hosszúságú karaktersorozatot ad, amely az i kifejezés aktuális értékének megfelelő kódú karakterből áll. Az i kifejezés aktuális értékének a 0...255 tartományba kell esnie. A CHR\$ függvényt általában arra használjuk, hogy segítségével a képernyőre, nyomtatóra, stb. egy speciális karaktert küldjünk.

CINT(x)

- A megadott x kifejezés aktuális értékét egész számmá alakítja, és eredményül ezt az egész értéket adja. Ha az x kifejezés aktuális értéke nem esik a $-32768...32767$ intervallumba, a PRIMO OV - Overflow, Túlcsordulás - hibát jelez.

COS(x)

- Az x kifejezés radiánban értelmezett értékének koszinuszát szolgáltatja. Az eredmény minden esetben egyszerespontos valós szám lesz.

CSNG(x)

- A megadott x kifejezés aktuális értékét egyszerespontos valós számmá alakítja, és eredményül ezt a számértéket szolgáltatja.

ERL/ERR

- Valahányszor a PRIMO BASIC Interpreter a BASIC program, vagy parancs végrehajtása során egy hibát észlel, beállítja az ERL és ERR változók értékét. Az ERL változó értéke a hibás utasítást tartalmazó BASIC programsor sorszáma, az ERR változó értéke a hibára jellemző hibakód lesz. (Az egyes hibákhoz rendelt hibakódok a Függelékben találhatók.)

Az ERL és ERR változókat általában programozott hibakezelést végző rutinban alkalmazzuk. Segítségükkel megállapítható, hogy a BASIC program melyik részén, milyen hiba fordult elő.

Ha a hibát okozó utasítást parancs módban adtuk meg, az ERL változó értéke 65535 lesz. Így a programozott hibakezelést végző rutint úgy is kialakíthatjuk, hogy parancs módban felismert hibák esetén azok elhárításának megkísérlése helyett, jelenítse meg a hibajelzést. Egy ilyen hibakezelő rutin első utasítása pl. a következő alakú lehet:

```
105 IF ERL=65535 THEN ON ERROR GOTO 0
```

(Ha a hibarutinban kikapcsoljuk a hibakezelést, megjelenik a hibarutinba ugrást kiváltó hibára utaló jelzés. Így a fenti utasítás a kitűzött célt megvalósítja. Ne felejtsük azonban el, hogy ezután a programozott hibakezelés csak akkor válik ismét aktív, ha azt az ON ERROR GOTO sorszám utasítással ismét bekapcsoljuk.)

Mivel az ERL és ERR változók BASIC kulcsszavak, ezért programozottan nem kaphatnak értéket. Ha mégis megkísérlünk e

----- PRIMO BASIC NYELV LEÍRAS -----

változóknak értéket adni, SN - Syntax Error, Formai hiba - hibajelzés keletkezik.

EXP(x)

- Eredményül az e^x értéket adja. Az x kifejezés aktuális értéke nem haladhatja meg a 87.33655 értéket, ellenkező esetben OV - Overflow, Túlszordulás - hiba lép fel. Az EXP függvény által szolgáltatott eredmény minden esetben egyszeres pontos valós szám lesz.

FIX(x)

- Az x kifejezés aktuális értékének egészrészét adja eredményül. A függvényérték kiszámítása a következő képlet alapján történik:

$$\text{FIX}(x) = \text{SGN}(x) * \text{INT}(\text{ABS}(x))$$

A FIX és INT függvények közti különbség abban áll, hogy a FIX negatív argumentum esetén nem a szomszédos kisebb, hanem a nagyobb értéket szolgáltatja.

Példák: PRINT FIX(7.46)
 7

 PRINT FIX(-7.46)
 -7

FRE(0)

FRE("")

- A függvény eredményül a BASIC által még fel nem használt memória byte-ban adott méretét szolgáltatja.

A FRE argumentumát csak formai okokból kell megadni, annak értéke közömbös, csak típusa lényeges.

Ha a FRE argumentuma numerikus, a függvény megadja, hogy a memóriában még hány byte használható fel a BASIC program és változói tárolására. Ha az argumentum karakteres típusú, a karakterület még fel nem használt részének byte-ban mért méretét kapjuk meg.

INKEY\$

- Megvizsgálja, hogy a billentyűzet melyik gombját nyomják. Ha egyetlen billentyűt sem érintünk, az INKEY\$ eredményül 0 hosszúságú karaktersorozat - üres stringet - ad. Ha nyomjuk a klaviatúra valamelyik gombját, a keletkező - egy karakter hosszú - karaktersorozat a megérintett gomb ASCII kódját tartalmazza.

Az INKEY\$ a billentyűzeten megnyomott gomb képét nem írja ki

a képernyőre. Így e függvényt többnyire akkor alkalmazzuk, ha a BASIC program által feltett kérdésre várjuk az egyetlen karakterből álló választ, vagy arra várunk, hogy a Felhasználó valamely billentyű megnyomásával jelezze, hogy a BASIC program végrehajtása folytatódhat.

INP(*i*)

- Az INP függvény segítségével a Z80 mikroprocesszor - az *i* kifejezés aktuális értéke által - megcímzett input portjáról olvashatunk be egy 8 bites adatot. A port-cimnek a 0...255 tartományba kell esnie.

INT(*x*)

- Eredményül az *x* kifejezés aktuális értékéhez legközelebb eső, annál kisebb, vagy azzal egyenlő egész számot adja.

Példák: PRINT INT(7.46)
7

PRINT INT(-7.46)
-8

LEFT\$(*x\$,i*)

- Az *x\$* karakteres kifejezés által adott karaktersorozat bal-oldalán elhelyezkedő, *i* karakter hosszúságú stringet adja eredményül. *i* értékének a 0...255 tartományba kell esnie. Ha *i* aktuális értéke nagyobb, mint *x\$* hossza, az eredmény a teljes *x\$* karaktersorozat lesz. Ha *i*=0, üres string lesz az eredmény.

LEN(*x\$*)

- Az *x\$* karaktersorozatot alkotó karakterek számát - a string hosszát - adja. A hosszba a képernyőn meg nem jelenő, speciális karakterek is beleszámítanak. Az üres string hossza 0.

LOG(*x*)

- *x* kifejezés aktuális értékének természetes alapú logaritmusát - $\ln x$ - adja eredményül. *x* értékének 0-nál nagyobbnak kell lennie. Az eredmény minden esetben egyszerespontos valós szám lesz.

MID\$(*x\$,i[,j]*)

- Az *x\$* karaktersorozat *i*. karakterétől kezdődő, *j* karakter hosszúságú stringet eredményez. Az *i* kifejezés aktuális értékének az 1...255 intervallumba, *j* értékének - ha megadjuk - a 0...255 tartományba kell esnie. Ha a *j* kifejezést nem adjuk meg, vagy *j* értékénél kevesebb karakter van az *x\$*

----- PRIMO BASIC NYELV LEÍRAS -----

string *i*. karakterétől jobbra, az eredménystring az *i*. karaktertől kezdődő, valamennyi jobbra eső karaktert tartalmazni fogja. Ha *i* értéke nagyobb, mint *x\$* hossza, eredményül üres stringet kapunk.

PEEK(*i*)

- Eredményül az *i* kifejezés aktuális értéke által megcímzett memóriabyte értékét adja. Az eredmény egy, a 0...255 tartományba eső egész szám lesz. A PEEK függvény argumentumaként megadott értéket a POKE utasításnál leírtak szerint kell megadni. A PEEK függvény a POKE utasítás ellentétének felel meg.

PI

- A PI a Ludolf-féle számot - 3.14159 - jelképezi. Mivel a PI BASIC kulcsszó, ezért programozottan nem kaphat értéket. Ha mégis megkíséreljük az értékadást, SN - Syntax Error, Formai hiba - hibajelzés keletkezik.

POINT(*i*,*j*)

- E függvény segítségével a PRIMO képernyője egy pontjának állapotát ellenőrizhetjük. Az *i* kifejezés a pont x-koordinátáját adja, értékének a 0...255 intervallumba kell esnie. A *j* a pont y-koordinátáját határozza meg, ennek értéke a 0...191 tartományban lehet.

A POINT függvény eredményül a 0, vagy a -1 egész számot adja. Ha a megcímzett pont világít - fekete alapon fehér, vagy fehér alapon fekete - az eredmény -1 lesz. Ha a vizsgált pont színe megegyezik a képernyő aktuális alapszínével, az eredmény 0 lesz.

POS(0)

- A képernyőn elhelyezkedő kurzor aktuális oszlopcímét adja meg. A bal szélső oszlop címe 0. Mivel a PRIMO egy képernyősorban 42 karaktert képes megjeleníteni, ezért a POS függvény által szolgáltatott egész szám a 0...41 tartományba esik. A függvény argumentumát csak formai okokból kell megadni, értéke közömbös.

RIGHT\$(*x\$*,*i*)

- Az *x\$* karakteres kifejezés által adott karaktersorozat jobb-
oldalán elhelyezkedő, *i* karakter hosszúságú stringet adja
eredményül. *i* értékének a 0...255 tartományba kell esnie. Ha
i aktuális értéke nagyobb, vagy egyenlő *x\$* hosszával, az

eredmény a teljes x\$ karaktersorozat lesz. Ha i=0, üres string lesz az eredmény.

RND(x)

- Az álvéletlenszám sorozat következő elemét generálja. A PRIMO az x kifejezés egész értékét veszi figyelembe.

Ha az x kifejezés aktuális értéke negatív, FC - Illegal Function Call, Illegális funkcióhívás - hiba keletkezik.

Ha az x kifejezés aktuális értéke $0 \leq x < 1$, egy $0 \leq \text{RND}(x) < 1$ intervallumba eső véletlenszám keletkezik.

Ha az x kifejezés aktuális értéke nagyobb, vagy egyenlő 2-vel, a PRIMO a generált véletlen számot megszorozza az x kifejezés aktuális értékével, és az így kapott szám egész részénél eggyel nagyobb számot adja eredményül, vagyis egy $0 \leq \text{RND}(x) \leq x$ intervallumba eső véletlen egész számot generál.

Ha az x kifejezés aktuális értéke $1 \leq x < 2$, véletlen számként a PRIMO konstans 1 értéket szolgáltat.

SGN(x)

- Az x kifejezés aktuális értéke előjelének függvényében -1, 0, +1 értéket ad eredményül. Az eredmény a következőképpen alakul:

Ha $x < 0$, SGN(x) = -1
 Ha $x = 0$, SGN(x) = 0
 Ha $x > 0$, SGN(x) = +1

SIN(x)

- Az x kifejezés radiánban értelmezett értékének szinuszt szolgáltatja. Az eredmény minden esetben egyszerespontos valós szám lesz.

SQR(x)

- Az x kifejezés aktuális értékének négyzetgyökét adja. Az eredmény minden esetben egyszerespontos valós számérték lesz. x aktuális értékének pozitívnak kell lennie.

STR\$(x)

- Eredményül egy karaktersorozatot ad, amely x kifejezés aktuális számértékének karakteres megfelelője lesz. Az STR függvény segítségével tehát egy numerikus értéket karaktersoro-

----- PRIMO BASIC NYELV LEÍRAS -----

zattá konvertálhatunk. Az eredményül kapott karaktersorozat azonos azon karakterek sorozatával, amelyek ugyanezen x érték PRINT utasítással való kiírása során megjelennének.

STRING\$(i, j, x)

- A függvény egy i karakter hosszúságú karaktersorozatot eredményez, amelynek összes karaktere azonos lesz. A keletkező stringet alkotó karakterek kódját a függvény második operandusa határozza meg.

Ha második operandusként numerikus kifejezést - j - adunk meg, a string e kifejezés aktuális értéke által meghatározott ASCII kódú karakterből fog állni. Ha a kifejezés karakteres értéket szolgáltat - x -, a stringet alkotó karakterek az x karaktersorozat első karakterével lesznek azonosak. Az i és j kifejezések aktuális értékének a 0...255 intervallumba kell esnie.

TAB(i)

- A TAB függvény csak a PRINT és LPRINT utasításokban alkalmazható. Segítségével a kurzor a soron belül a megadott sorszámú oszlopig előre mozgatható. Az i kifejezés aktuális értékének PRINT utasítás esetén a 0...41 intervallumba, LPRINT utasításban a 0...255 tartományba kell esnie. Ha a kurzor már túlhaladt a megcímzett oszlopon, a TAB függvény hatástalan.

TAN(x)

- Az x kifejezés radiánban értelmezett értékének tangensét szolgáltatja. Az eredmény minden esetben egyszerespontos valós szám lesz.

VAL(x)

- Az x karakteres kifejezés által adott szám numerikus értékét adja meg. A VAL függvény az STR\$ függvény ellentétének tekinthető.

A PRIMO az x kifejezés karaktereit addig értelmezi, amíg azok egy szám karaktereinek tekinthetők. Ha a string feldolgozása közben egy számkonstansban meg nem engedett karaktert talál, befejezi az értelmezést, és az addig kapott numerikus értéket adja eredményként. Ha a karaktersorozatban a PRIMO nem talál értelmezhető számot, a VAL függvény eredményül 0-t ad. A szám egész-, fixpontos-, vagy lebegőpontos formában adható meg. Ha a szám előtt betűköz karakterek állnak, azokat a PRIMO figyelmen kívül hagyja.

Példa: PRINT VAL(" -8.439 itt a vége")
-8.439

VARPTR((változó))

- Eredményül a megadott változónév értékezejének memórián belüli kezdőcímét adja. A változó bármely típusú skalár vagy tömbváltozó lehet, de a VARPTR függvény argumentumában való megjelenést megelőzően már értéket kellett kapjon, ellenkező esetben FC - Illegal Function Call, Illegális funkcióhívás - hibajelzés keletkezik. Az eredményként keletkező egész szám a -32768...32767 tartományba esik. Ha az érték negatív, a valódi memóriacím 65536 hozzáadásával állítható elő.

A VARPTR függvényt általában arra használjuk, hogy megszerezve egy változó memóriacímét, e változót - címének átadásával - paraméterként egy gépi kódú rutin rendelkezésére bocsássuk. Ha változóként egy indexelt változót adunk meg, megtudhatjuk a kérdéses tömbelem címét, vagy - ha a tömb összes indexe értékeként 0-át szerepeltetünk - a teljes tömb kezdőcímét.

Egy tömb memóriacíme minden egyes - eddig értéket még nem kapott - skalár változó használatakor megváltozik. Ezért, ha a VARPTR függvénnyel egy tömb memóriacímét kívánjuk megszerezni, ügyeljünk arra, hogy újabb - eddig még nem használt - skalár változókat ezután már ne használjunk, mert ez programunk helytelen működését eredményezheti.

2. RÉSZ

A PRIMO számítógép működése

1. Fejezet

A PRIMO BASIC Interpreter működése

1.1. A BASIC program memórián belüli ábrázolása	II-1
1.2. A BASIC Interpreter működése	II-4
1.3. A BASIC változók táblázatai	II-6
1.4. Karakteres változók kezelése	II-12
1.5. STACK bejegyzések	II-15
1.6. Kifejezések feldolgozása	II-19
1.7. Területfoglalás gépi kódú rutinok számára	II-24

2. Fejezet

A PRIMO aritmetikai rutinjai

2.1. A PRIMO számábrázolási formátumai	II-29
2.2. Az aritmetika munkaregiszterei	II-36
2.3. A Karakteres $\leftrightarrow$ Bináris konverzió rutinjai	II-37
2.4. Aritmetikai műveletek rutinjai	II-39
2.4.1. Egész típusú műveletek	II-39
2.4.2. Egyszeres pontos valós műveletek	II-41
2.4.3. Duplapontos valós műveletek	II-42
2.5. Függvények	II-43

3. Fejezet

A PRIMO periféria kezelő rutinjai

3.1. A periféria kezelő rutinok ugrótáblája	II-47
3.2. NMI/TIMER rutin	II-48
3.3. RESET rutin	II-48
3.4. Várakozás képkioztásra rutin - VBLANK	II-49
3.5. Képernyő kezelő rutin - DSPHND	II-49
3.6. Billentyűzet kezelő rutinok	II-51
3.7. Nyomtató kezelő rutin - PRTHND	II-52
3.8. Egy sort beolvasó rutin - GLINE	II-53
3.9. Magnó kezelő rutinok	II-55
3.9.1. Logikai állománykezelés	II-55
3.9.2. A magnó fizikai kezelése	II-59
3.9.3. Magnó kezelő szubrutinok	II-63
3.10. Direkt kurzor címző rutin - DIRCUR	II-65
3.11. Kurzor pozíciót megállapító rutin - KURPOZ	II-66
3.12. Grafikus pont kezelő rutinok	II-66

----- FELJEGYZÉSEK -----

1. FEJEZET

A PRIMO BASIC Interpreter működése

Ebben a fejezetben összefoglaljuk a PRIMO BASIC Interpreter működésének megértéséhez szükséges legfontosabb ismereteket. Bemutatjuk a BASIC program memórián belüli ábrázolásának módját, ismertetjük a BASIC program végrehajtásának főbb lépéseit. Megadjuk a BASIC program végrehajtása során keletkező változó táblázatok, STACK bejegyzések pontos felépítését, ismertetjük a karakteres változók feldolgozásának lépéseit, kitérünk a kifejezések feldolgozásának menetére. Végezetül rövid példaprogramok segítségével bemutatjuk, hogyan tud a Felhasználó saját gépi kódú programjai számára területet foglalni a memóriában.

1.1. A BASIC program memórián belüli ábrázolása

A PRIMO BASIC Interpretere a BASIC program gyorsabb és egyszerűbb végrehajtása érdekében a billentyűzetről begépelte BASIC programszöveget a memóriában átalakítva - u.n. tokenizált formában - tárolja. Az ábrázolási mód lényege, hogy a programszövegben a BASIC programozási nyelv kulcsszavai - pl. GOTO, PRINT, stb. - és az operátorok - pl. +, -, =, stb. - nem eredeti formájukban, hanem átalakítva kerülnek tárolásra. Az átalakítás során az említett szimbólumokat az Interpreter egy 1 byte hosszúságú belső kóddal - u.n. tokennel - helyettesíti. A megoldás két szempontból is előnyös.

- Az eredeti alakban adott kulcsszónál a token az esetek döntő többségében rövidebb, így az átalakítás eredményeként a BASIC programszöveg tárolásához kevesebb memória szükséges.
- Az ismert méretű token egyszerűbben kezelhető, mint a több karakter hosszúságú kulcsszó, így számos esetben a kulcsszavak feldolgozásának sebessége - végső soron a BASIC program végrehajtása - jelentősen gyorsabban valósítható meg az átkódolt formában.

A kulcsszavak tokenekre cserélésén túl az Interpreter az eredetileg begépelte BASIC programszövegen még további átalakításokat is végez. Ezek ismét csak a végrehajtás sebességének fokozását célozzák, de már nem jelentenek memória megtakarítást, ellenkezőleg, a BASIC programszöveg tárolásához szükséges tárterület méretének kismértékű megnövekedésével járnak. (A méretnövekedés nem jelentős, mivel legrosszabb esetben egy BASIC programsor tokeni-

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

zálás utáni méretét 4 byte-tal növeli meg.) A további átalakítások a következők:

- Az Interpreter a BASIC programszöveg sorait nem a begépelés sorrendjében, hanem az egyes sorok elején álló sorszámok növekvő sorrendjében tárolja. Így aztán amikor egy újabb programsort fűzünk a memóriában eddig eltárolt BASIC programhoz, gyakran szükséges a programszöveg egyes részeinek memórián belüli mozgatása annak érdekében, hogy az újonnan begépelte programsor a helyére kerülhessen. A fenti tevékenység - valamint más, a BASIC program végrehajtása során szükségessé, itt nem részletezett feladatok - hatékonyabb elvégzése érdekében az Interpreter minden BASIC programsor elején kialakít egy 2 byte-os mezőt, amelyben az aktuális programsort követő sor hasonló mezejének memória címét tárolja.
- A BASIC program végrehajtása során igen gyakran szükséges az adott programsor sorszámának ismerete. Azért, hogy az Interpreter ne kényszerüljön a karakteres alakban megadott sorszám ismételt Karakteres → Bináris konverziójára, minden BASIC programsor elején - az előbb ismertetett cím-mezőt követő - 2 byte-ban a PRIMO eltárolja az adott BASIC programsor sorszámának bináris megfelelőjét.
- A végrehajtás alatt álló BASIC programsor végének egyszerű felismerése érdekében az Interpreter minden programsor utolsó karaktere után elhelyez egy 0 értékű byte-ot. Így, ha egy sor következő karaktere 0, ez jelzi az Interpreter számára, hogy az adott programsor feldolgozása befejeződött, megkezdheti a soronkövetkező BASIC programsor értelmezését. Ha azonban a 0 értékű byte-ot két további 0 követi, akkor ebből a PRIMO Interpretere felismeri, hogy elért a BASIC program végére. Ezután - további végrehajtandó programsorok hiányában - visszatér direkt - parancs - módba, és a Felhasználó újabb parancsára várakozik.

Az eddig ismertetett átalakításokon túl a PRIMO BASIC Interpretere a begépelte BASIC programsoron semmiféle egyéb átalakítást nem végez. Így pl. ha a begépelte programsor az olvashatóbb formátumra törekvés érdekében betűközöket tartalmaz, akkor azok változatlan formában a programszövegben is megtalálhatók. (Az egyedüli kivétel a programsor sorszáma és a programsor első nem betűköz karaktere között álló betűközök. Ezeket ugyanis a PRIMO minden esetben figyelmen kívül hagyja és el sem tárolja.) Ez az Interpreter működését nem befolyásolja, mivel az a BASIC program értelmezése közben képes a felesleges betűközök átlépésére. Ugyanakkor azonban az egyszerűbb programolvasás ára a nagyobb memória igény és a BASIC program végrehajtásának lassulása lesz. Az utóbbi szempont gyakorlatilag lényegtelen, mivel a sebességcsökkenés szinte észrevehetetlen. A szükséges memória méretének

----- A PRIMO SZÁMÍTÓGÉP MÓKODÉSE -----

növekedése már több figyelmet érdemel. Ha a BASIC programunk olyan nagy méretű, hogy tárolása és futtatása már gondot okoz - nevezetesen OM - Out of Memory, Betelt a memória - hibajelzést kapunk, akkor az egyébként felesleges betűkötők kitörölésével esetleg még megoldható a program futtatása.

A következőkben az eddig leírtakat egy ábrán szemléltetjük. Két egyszerű BASIC programsor példáján bemutatjuk, hogyan tárolja a PRIMO a BASIC programszöveget.

Legyenek a BASIC utasítások a következők:

```
1000 X=2*PI-LOG(A)
1010 GOTO 200
```

Tételezzük fel, hogy ez a két utasítás egy nagyobb BASIC program utolsó két utasítása. Tegyük fel továbbá, hogy az 1000-es sorszámú sor a memóriában a 20480;5000 címűl kezdve tárolódik. (Az ábrában szereplő valamennyi kód értéket decimális és hexadecimális alakban kifejezve adjuk meg.)

20480;5000	15;0F	Következő sor cím	alsó byte
20481;5001	80;50		felső byte
20482;5002	232;E8	BASIC sor sorszáma	alsó byte
20483;5003	03;03		felső byte
20484;5004	X 88;58		
20485;5005	= 213;D5	Token	
20486;5006	2 50;32		
20487;5007	* 207;CF	Token	
20488;5008	PI 200;CB	Token	
20489;5009	- 206;CE	Token	
20490;500A	LOG 223;DF	Token	
20491;500B	(40;28		
20492;500C	A 56;41		
20493;500D	) 41;29		
20494;500E	00;00	Sorvég	
20495;500F	25;19	Következő sor cím	alsó byte
20496;5010	80;50		felső byte
20497;5011	242;F2	BASIC sor sorszáma	alsó byte
20498;5012	03;03		felső byte
20499;5013	GOTO 141;8D	Token	
20500;5014	32;20	betűköz	
20501;5015	2 50;32		
20502;5016	0 48;30		
20503;5017	0 48;30		
20504;5018	00;00	Sorvég	
20505;5019	00;00	Programvég jel #1	
20506;501A	00;00	jel #2	

----- A PRIMO SZÁMÍTÓGÉP MŰKÖDÉSE -----

1.2. A BASIC Interpreter működése

A PRIMO bekapcsolása és a szükséges kezdeti beállítások után a számítógép vezérlését a BASIC Interpreter veszi át. A képernyő legfelső sorában megjeleníti a

PRIMO BASIC SYSTEM '84.1

feliratot, majd a gép direkt - parancs - módban a Felhasználó parancsaira várakozik. Ennek során folyamatosan vizsgálja a billentyűzetet, és ha egy billentyű megnyomását érzékeli, a gomb által jelképezett karakter kódját a Billentyűzet puffer következő szabad címére helyezi. Az Interpreter a leírt tevékenységet addig ismétli, amíg a BRK, ill. RETURN billentyű megnyomását nem tapasztalja. Ekkor az események a következőképpen alakulnak:

- Ha a BRK billentyűt érintettük meg, az Interpreter az eddig a Billentyűzet pufferbe tárolt kódokat "elfelejti", majd egy újabb sor begépelésére várakozik.
- Ha a RETURN billentyűt nyomtuk meg, de előtte - a betűköztől különböző - más gombot még nem, az Interpreter ugyanúgy jár el, mintha a BRK lett volna aktiv. Ha a RETURN megérintése előtt már begépelünk egy - maximálisan 210 karakter hosszúságú - sort, akkor megkezdődik ennek a sornak a feldolgozása.

Egy sor feldolgozása azzal kezdődik, hogy a PRIMO átalakítja a begépelte sort. Ennek során a Billentyűzet pufferben tárolt karaktersorozatban található valamennyi BASIC kulcsszót és műveleti jelet a megfelelő belső kóddal - tokennel - helyettesíti. Ezután attól függően, hogy a sor elején állt-e sorszám vagy sem, a PRIMO különböző módon folytatja a sor további feldolgozását.

Ha a sor elején állt sorszám, a további működés menete a következő:

- Az Interpreter megvizsgálja, hogy követik-e további karakterek is a sorszámot. Ha nem - vagyis a sor csak egy sorszámot tartalmaz - a PRIMO az eddig begépelte és a memóriában őrzött sorok közül - ha van ilyen - kitörli a megadott sorszámú sort, majd egy újabb sor begépelésére várakozik.
- Ha a sorszám után további karakterek is találhatók, a PRIMO megvizsgálja, hogy a megadott sorszámmal van-e már a memóriában programsor. Ha talál ilyet, akkor a régi sort a most begépelte új sorral cseréli fel. Ha ilyen sorszámú sor még nem létezett, akkor a most begépelte sort tárolja el, majd egy újabb sor begépelésére várakozik.

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

Akár sor törlést, akár sor beillesztést kell végrehajtani, szükségessé válik a memóriában őrzött **BASIC** programszöveg tárolón belüli mozgatása. Ennek során a **PRIMO** a következő műveleteket hajtja végre:

- Meghatározza a **BASIC** programszöveg azon pontját, amelytől kezdődően - a tároló magasabb címeinek irányába - a memóriaterület elmozdítása szükséges.
- Ha sor törlést kell végrehajtani, a **PRIMO** az eltávolítandó sort követő programszöveget a memóriában a feleslegessé vált sor által eddig elfoglalt byte-ok számával a tároló kisebb címeinek irányába előretolja.
- Ha sor beillesztés a feladat, az Interpreter az új sort a sorszámok sorrendjében követő programsortól kezdődően az új sor hosszának megfelelő byte számmal a tároló magasabb címeinek irányába hátrátolja. Ezután a Billentyűzet pufferből az új sort a keletkezett szabad területre másolja.
- A **BASIC** programszöveg kiválasztott részének elmozdítása után az Interpreter a programsorok elején álló - a következő sor kezdőcímét tartalmazó - mezőket az új helyzetnek megfelelően beállítja.

Ha a sor elején nem áll sorszám, vagyis a begépelt sor egy **BASIC** parancsot, ill. egy parancs módban megadott **BASIC** utasítást tartalmaz, a működés további menete a következő:

- A **PRIMO** megvizsgálja, hogy a sorban álló első - betűköztől különböző - karakter token-e. Amennyiben nem, az Interpreter úgy tekinti, hogy az egy **BASIC** változó 1. karaktere. Mivel utasítás elején változó csak az értékadás műveletben szerepelhet, a **PRIMO** szubrutinként aktivizálja az értékadást végrehajtó rutint. A sor további feldolgozását már ez fogja végrehajtani.
- Ha a begépelt, és belső ábrázolási módra átalakított sor első - betűköztől különböző - karaktere egy token, a **PRIMO BASIC** Interpretere egy - ROM-területen elhelyezkedő - táblázat felhasználásával megállapítja az adott **BASIC** utasítás, ill. parancs értelmezésére és végrehajtására hivatott alprogram címét. A cím meghatározása után a **PRIMO** megkeresi a sorban következő első nem betűköz karaktert, majd miután ennek memóriacímét elhelyezte a HL regiszterpárban, szubrutinként aktivizálja a sor további értelmezését végrehajtó alprogramot.

Az egyes utasítások feldolgozását végző szubrutinok tevé-

----- A PRIMO SZAMITOGEP MOKODESE -----

kenységük befejeztével egy RET utasítás segítségével léphetnek vissza az Interpreter Központi vezérlő részébe. Miután e Központi vezérlő rész ismét megkapja a PRIMO irányítását, az eddig leírtak egy újabb sor bevételeivel, értelmezésével, stb. ismétlődnek.

A parancsok közül kettő - a RUN és a CONT - kitüntetett jelentőségűek. Ezek végrehajtása után ugyanis a PRIMO irányítását nem az Interpreter Központi vezérlő része kapja vissza, hanem a memóriában őrzött BASIC program. Ez az állapot mindaddig nem is változik meg, amíg a PRIMO el nem ér a tárolt BASIC program végére, ill. végre nem hajt egy STOP vagy END utasítást. Ekkor ismét az Interpreter Központi vezérlő része lép működésbe. (A PRIMO hátoldalán elhelyezett RESET billentyű megnyomása is megszakítja a program futását, és az irányítást az Interpreter Központi vezérlő része veszi át.)

A RUN parancs végrehajtásakor a BASIC program elindításán túl az Interpreter még számos feladatot hajt végre. A következőkben ezeket a tevékenységeket foglaljuk össze.

Valahányszor begépeljük a RUN parancsot, a PRIMO

- a különféle betűkkel kezdődő változók típusát meghatározó táblázat minden elemébe egyszerespontos valós típusjelzést helyez. Ennek megfelelően a BASIC program mindazon változójának típusa - amely DEFxxx utasítások, ill. típusazonosító karakterek segítségével ellenkező értelmezést nem kap - egyszerespontos valós lesz.
- alapállapotba állítja a "Programozott hibakezelés folyamatban" állapotjelzőt.
- törli valamennyi korábban értéket kapott skalár és indexelt, numerikus és karakteres változó értékét.

A fenti feladatok végrehajtása után megkezdődik a PRIMO memóriájában tárolt BASIC program végrehajtása.

(Az Interpreter működésének a fentieknél részletesebb - pl. az egyes BASIC utasítások és parancsok értelmezését végrehajtó rutinok részletes - ismertetése meghaladná a rendelkezésre álló terjedelmet.)

1.3. A BASIC változók táblázatai

A PRIMO a BASIC program végrehajtása közben felismert változókat az u.n. változó táblázatokban őrzi. Egy változó attól függően, hogy skalár, ill. tömb változó-e, különböző formában kerül a memóriába. A BASIC Interpreter működése során két változó

táblázatot; a skalár változók és az indexelt - tömb - változók táblázatát építi fel. (A PRIMO a numerikus változóktól eltérő módon dolgozza fel a karakteres változókat. Ez utóbbiak kezelésének részletei az 1.4. Fejezetben találhatók.)

Amikor a PRIMO megkezdi a memóriában tárolt BASIC program végrehajtását, a változó táblázatok még üresek. Minden alkalommal, amikor az Interpreter a BASIC programszövegben egy újabb változót ismer fel, megvizsgálja, hogy szerepel-e már a kérdéses változó a formájának - skalár, indexelt - megfelelő táblázatban. A válasz függvényében az Interpreter a következőképpen folytatja működését:

- Ha az adott változó még nem szerepel a formájának megfelelő táblázatban, akkor most az Interpreter beilleszti a megfelelő helyre. Ennek során a táblázatba kerül a változó típusa - egész típusú, egyszerespontos, ill. duplapontos valós, karakteres -, a változó maximálisan két karakter hosszúságú neve, valamint a változó által képviselt érték. Ha a változó indexelt, az Interpreter a felsorolt adatokon túl még az indexek számát, és az egyes indexek maximális értékét is a táblabejegyzésbe tárolja.
- Ha a változó már megtalálható a megfelelő táblázatban, akkor az Interpreter a végrehajtott BASIC utasítás függvényében vagy csak beírja - értékadás -, vagy csak kiolvassa a táblázatból a kérdéses változó értékét.

A változó táblázatok a memóriában a BASIC programszöveg mögött helyezkednek el. A memória kisebb címein a skalár változó tábla, ezt követően az indexelt változó tábla található. A BASIC program módosításakor, ill. a program újra indításakor a táblázatok mindig törlődnek. Ha egy új skalár változó kerül a skalár táblába, a változó számára szükséges hely előteremtése érdekében az indexelt változók táblázatát az Interpreter a memória magasabb címeinek irányába elmozdítja. Ezután a PRIMO az új változót a skalár tábla végére illeszti. (A leírtakból látható, hogy egy skalár változó tábla-címe a BASIC program végrehajtása során nem változik meg. Az indexelt változók címe azonban minden újabb skalár változó előfordulásával módosul. Ezt azért fontos megjegyezni, mert a VARPTR függvény - amely egy változó memória címét adja - alkalmazásakor e tény figyelmen kívül hagyása a BASIC program szándékunktól eltérő futását eredményezheti.)

A következőkben részletesen ismertetjük az egyes táblázatok különféle bejegyzéseinek felépítését, az egyes mezők értelmezését:

----- A PRIMO SZÁMÍTÓGÉP MŰKÖDÉSE -----

Skalár változó tábla bejegyzések

A skalár változó táblában a változók négy lehetséges típusának - egész, egyszerespontos, duplapontos, karakteres - megfelelően négy különféle bejegyzés található. A bejegyzések rendre 5, 7, 11, ill. 6 byte méretűek. Egy bejegyzés logikailag két részre - Név-mező, érték-mező - bontható. A Név-mező minden bejegyzésben azonos felépítésű.

- A Név-mező 1. byte-ja a változó típusára jellemző értéket tárolja. Ez egész típusú változó esetén 2, egyszerespontos valós változónál 4, duplapontos valós típus esetén 8, karakteres változónál 3. (Ez az érték tulajdonképpen az érték-mező byte-ban mért hosszát jelzi.) A Név-mező további két byte-ja a változó nevének első két karakterét tárolja. Ennek megfelelően ugyan megadhatunk két karakternél hosszabb változó-nevet is, de a PRIMO csak az első két karaktert tekinti értékesnek. Ha a változó neve csak egy karakter hosszú, az Interpreter a 2. karakter helyére 0 értéket tárol.
- Az érték-mező a négy lehetséges változó típusnak megfelelően négy féle lehet. Az érték-mező a Név-mezőben álló típusazonosító kód által meghatározott hosszúságú.

A következő ábrák a négy lehetséges skalár tábla bejegyzés felépítését szemléltetik. (Az érték-mezők felépítésével kapcsolatban részletes ismeretek az 1.4 és 2. Fejezetekben találhatók.)

Egész típusú tábla elem

Tipus kód (= 2)
Változónév 2. karaktere
Változónév 1. karaktere
érték kevésbé értékes 8 bitje
érték értékesebb 8 bitje

Egyszerespontos valós tábla elem

Típus kód (= 4)
Változónév 2. karaktere
Változónév 1. karaktere
Mantissza legkevésbé értékes 8 bitje
Mantissza közbűlső 8 bitje
Mantissza legértékesebb 8 bitje
Kitevő

Duplapontos valós tábla elem

Típus kód (= 8)
Változónév 2. karaktere
Változónév 1. karaktere
Mantissza legkevésbé értékes 8 bitje
Mantissza közbűlső 8 bitje #1
Mantissza közbűlső 8 bitje #2
Mantissza közbűlső 8 bitje #3
Mantissza közbűlső 8 bitje #4
Mantissza közbűlső 8 bitje #5
Mantissza legértékesebb 8 bitje
Kitevő

----- A PRIMO SZÁMITÓGÉP MÓKODÉSE -----

Karakteres típusú tábla elem

Tipus kód (= 3)
Változónév 2. karaktere
Változónév 1. karaktere
érték-hossz
érték cím kevésbé értékes 8 bitje
érték cím értékesebb 8 bitje

(A karakteres változók kezelésének részletei az 1.4. Fejezetben találhatók meg.)

Indexelt változó tábla bejegyzések

Akár a skalár változó táblában, az indexelt változó táblában is alapvetően négyfajta bejegyzés található. Azonban az indexek eltérő száma miatt az egyes bejegyzések felépítése különböző lehet. Az indexelt tábla egy bejegyzése logikailag három részre - Név-mező, Index-mező, érték-mező - bontható. A Név-mező és az érték-mező felépítése a skalár változó tábla leírásában megismert formátummal - eltekintve attól, hogy az érték-mező nem egyetlen, hanem a tömb valamennyi elemének értékét őrzi - azonos. Az Index-mező felépítése a következő:

- Mivel az indexelt változó tábla bejegyzései - a tömbelemek különböző száma miatt - eltérő hosszúságúak, a következő tábla bejegyzés elejének gyors megtalálása érdekében az Index-mező első két byte-ja az aktuális tábla bejegyzés hosszát őrzi. Így a következő tábla elem címe az aktuális elem címének és tárolt hosszának ismeretében egy összeadás-sal meghatározható.
- A PRIMO a tömbváltozók indexeinek számát gyakorlatilag nem korlátozza - elméletileg egy tömb 255 dimenziós lehet -, továbbá az Interpreter két azonos nevű tömböt különbözőnek tekint, ha indexeik száma eltérő. Ezért, szükséges, hogy a PRIMO az Index-mezőben eltárolja az aktuális tömb indexeinek számát. Így az Index-mező soronkövetkező byte-ja előjeltelen egész szám formájában ezt az értéket őrzi. (A tömb indexeinek számát a BASIC DIM utasításával határozhatjuk meg.)
- Az Index-mező hátralévő részében a PRIMO az egyes indexek maximális értékeit - pontosabban a maximális érték + 1-et -

----- A PRIMO SZÁMITÓGÉP MOKODÉSE -----

tárolja. Egy-egy index-érték tárolására 2 byte szolgál, amelyben az Interpreter az index-értéket előjeltelen egész szám formájában tárolja. Ennek megfelelően egy index maximális értéke elvileg 65535 lehet. (Gyakorlatilag ekkora érték a memória korlátozott mérete miatt nem adható meg.) A bejegyzésben tárolt index-értékek száma az indexek számával azonos.

A leírtak figyelembe vételével az Index-mező byte-ban mért hossza a következő képlet segítségével határozható meg:

$$\text{Index-mező hossza} = 3 + 2 * (\text{indexek száma})$$

Az indexelt változó bejegyzés érték-mezejében az indexek száma, és az egyes indexek maximális értéke által meghatározott számú egyedi érték tárolható. Egy-egy érték tárolása azonos a skalár változóknál megismerttel.

A PRIMO a tömb elemeit oszlopfolytonosan tárolja. Ez azt jelenti, hogy az indexek közül leggyorsabban az első - bal szélső - változik. Ennek megfelelően, ha pl. az A nevű tömb két dimenziós, és a baloldali index maximális értéke 2, a jobboldali index maximális értéke 1, akkor a tömb 6 eleme a következő sorrendben helyezkedik el az érték-mezőben: (Az indexek első értéke minden esetben a 0.)

A(0,0) A(1,0) A(2,0) A(0,1) A(1,1) A(2,1)

Mivel az érték-mezőben az egyes értékek között semmilyen elválasztó jel nem áll, a PRIMO az egyedi érték címét határozza meg. Ennek ismeretében már az érték kiolvasása/módosítása egyszerűen megoldható. Valahányszor egy érték egyedi címét meg kell határozni, az Interpreter kiszámolja az u.n. címfüggvény aktuális értékét. A kiszámolandó kifejezés a következő alakú:

$$C = KC + H * (...(((I_n * M_{n-1} + I_{n-1}) * M_{n-2} + I_{n-2}) * M_{n-3} + I_{n-3}) * ... * M_1 + I_1)$$

ahol C = a keresett érték címe

KC = az érték-mező kezdőcíme

H = az értékmezőben tárolt értékek hossza (2,3,4,8)

I = az aktuális index-érték

M = a maximális index-érték+1

n = az indexek maximális száma

Pl. tekintsük az X egyszeresponstos valós tömböt. A tömb három dimenziós, a maximális index-értékek rendre 5,3,6. A tömb érték-mezejének memórián belüli kezdőcíme 23471. Keressük az X(3,1,4) tömbelem címét:

$$C = 23471 + 4 * ((4 * 4 + 1) * 6 + 3) \rightarrow 23891$$

----- A PRIMO SZAMITOGEP MOKODESE -----

Az eddig leírtak illusztrálására a következő ábra bemutatja az előző példában szereplő X tömb Index-mezejének felépítését, az egyes mezők - választott példának megfelelő - értékeit:

érték-mező hossz kevésbé értékes 8 bitje	= 160
érték-mező hossz értékesebb 8 bitje	= 2
Maximális indexszám	= 3
1. index maximális értéke + 1	= 6
2. index maximális értéke + 1	= 0
3. index maximális értéke + 1	= 4
	= 0
	= 7
	= 0

1.4. Karakteres változók kezelése

Ellentétben a különböző típusú numerikus változókkal, melyek tárolásához minden esetben rögzített méretű memóriahely szükséges, a karakteres változók a bennük pillanatnyilag tárolt karaktersorozat hosszának függvényében változó méretű memóriaterületet igényelnek. E sajátosság következtében a karakteres változók a numerikus változók feldolgozásától jelentősen különböző kezelést igényelnek. A következőkben áttekintjük a karakteres változók memórián belüli tárolásával, és feldolgozásával kapcsolatos fontosabb ismereteket.

A PRIMO BASIC Interpretere 0...255 karakter hosszúságú karaktersorozatokat képes kezelni. Egy karakteres változó aktuális értéke is hasonló hosszúságú lehet. Mivel azonban a karaktersorozat hosszának változása függvényében más és más méretű memória szükséges a változó tárolásához, a PRIMO a karakteres változókat két részletben - Leíró-rész, érték-rész - tárolja. E megoldás előnye, hogy egy karakteres változó mindig csak az aktuális értéke által pillanatnyilag szükséges memóriaterületet foglalja el. Az alkalmazott megoldás hátránya, hogy némileg lassabb végrehajtási sebességet eredményez, mint a memóriával kevésbé takarékoskodó módszerek.

A karakteres változót leíró táblaelem tehát két részből áll. A két rész nemcsak logikailag válik szét, hanem a memóriában is

különböző területeken helyezkedik el. A Leiró-rész - amely minden esetben 3 byte méretű - a skalár-, ill. - ha a vizsgált karakteres változó tömbváltozó - az indexelt változó táblában található.

A karakteres változó Leiró-része logikailag további két részre - érték-hossz, érték-cím - bontható. Az érték-hossz 1 byte méretű, az érték-cím tárolásához 1 szó - 2 byte - szükséges. Az egyes részek jelentése a következő:

- Az érték-hossz őrzi a karakteres változó aktuális értékének pillanatnyi hosszát. Az 1 byte-os mezőben a PRIMO egy elő-jeltelen egész számot tárol. Ennek megfelelően a mező értéke 0...255 között változhat. A mező 1 byte-os méretéből következik, hogy a PRIMO nem képes 255 karakternél hosszabb karaktersorozatokat feldolgozni. Ha a karakterváltozó aktuális értéke egy üres karaktersorozat - amely egyetlen karaktert sem tartalmaz -, az érték-hosszban 0 található.
- Az érték-cím a karakterváltozó aktuális értékét jelképező karaktersorozat memóriacímét őrzi. Ez egy 2 byte-os mező, tehát a karaktersorozatot a PRIMO memóriájában bárhol elhelyezhetjük, a rendelkezésre álló 16 bit elegendő a cím tárolásához. Ha a Leiró-rész érték-hossz mezőjének aktuális értéke 0, az érték-cím tartalma érdektelen.

A karakteres változó Leiró-részének felépítése tehát a következő:

érték-hossz
érték-cím kevésbé értékes 8 bitje
érték-cím értékesebb 8 bitje

A karakteres változók érték-része a tulajdonképpeni tárolt érték. Az érték-rész olyan hosszú, mint a benne tárolt karaktersorozat. Attól függően, hogy a karakteres változó által pillanatnyilag őrzött karaktersorozat - másnéven string - egy a BASIC programban megadott karakteres konstans, vagy egy korábban kiértékelt karakteres kifejezés eredménye, az érték-rész a memória különböző területein található.

- Ha a tárolt string egy karakterkonstans, akkor az érték-rész magában a BASIC programszövegben található. Így nem szükséges egy karakterkonstanst két helyen - a BASIC programban, ill. a változóterületen - őrzeni. Ez jelentős memória megtakarítást és végrehajtási sebesség növekedést eredményezhet.
- Abban az esetben, ha a tárolt string egy korábban kiértékelt

karakteres kifejezés eredménye, akkor az érték-rész a memóriában a karakteres változók érték-részeinek tárolására elkülönített területen - az u.n. Karakterváltozó területen - található. Ez a terület a Képernyő RAM és a STACK között helyezkedik el. A BASIC Interpreter a PRIMO bekapcsolásakor a Karakterváltozó területet 50 byte kiterjedésre definiálja. Ez azt jelenti, hogy a BASIC program által használt mindazon karakteres változó, amely pillanatnyilag nem karakterkonstans tárol, érték-részeinek együttes mérete nem haladhatja meg az 50 karakter hosszúságot. Ez már egyszerű programok esetén is kevésnek bizonyulhat, ezért lehetséges az említett terület méretének BASIC programból vezérelt megváltoztatása (növelése, vagy akár csökkentése is). Ezt a feladatot a BASIC CLEAR utasítása hajtja végre (ld. 1. RÉSZ 2.2. fejezet). Mivel a CLEAR utasítás valamennyi változó aktuális értékét törli, a Karakterváltozó terület, - méretének meghatározása után - mindaddig üres, amíg az első karakteres érték tárolása be nem következik.

Mint láttuk, a karakteres változók aktuális értéke - ha az nem egy karakterkonstans - a Karakterváltozó területen tárolódik. Amikor egy újabb karakteres változó értékét kell eltárolni, ill. egy korábban már felhasznált karakteres változó kap új értéket, a PRIMO BASIC Interpreter elhelyezi a stringet a Karakterváltozó területen. A terület feltöltése a kisebb memóriacímek irányából a nagyobbak felé halad. A következő egyszerű BASIC utasítás végrehajtásának elemzésével tekintsük át, hogyan is kezeli a PRIMO a Karakterváltozó területet.

Legyen a végrehajtandó utasítás a következő:

10 KSS=KSS+"A"

Az utasítás a KSS karakterváltozó korábbi értékéhez hozzáfűzi az "A" stringet - konkatenáció -, majd a változó új aktuális értéke az így keletkező karaktersorozat lesz.

A KSS karakterváltozó kezdeti értéke egy karakterkonstans, egy korábban végrehajtott karakteres kifejezés aktuális értéke, vagy az üres string lehet. Tételezzük fel, hogy KSS aktuális értéke egy karakteres kifejezés korábbi kiértékelésekor keletkezett. Ennek megfelelően az a Karakterváltozó területen található. A 10 sorszámú BASIC utasítás végrehajtásakor KSS korábbi értékénél eggyel hosszabb string keletkezik, így az nem tárolható az eredeti karaktersorozat helyén. Ezért a PRIMO megvizsgálja, hogy a Karakterváltozó területen rendelkezésre áll-e egy akkora memóriarész, amelyben az új string eltárolható.

- Ha rendelkezésre áll a szükséges hely, akkor az Interpreter betölti erre a területre a kiértékelés során keletkező új

stringet, majd a KSS változó leírójába elhelyezi az eredmény hosszát és címét. Az érték-cimben a karaktersorozat 1. karakterének memória címe található. Mivel a PRIMO a stringeket a Karakterváltozó területre növekvő címek irányába helyezi el, így a 2. karakter címét úgy kapjuk, hogy az 1. karakter címét eggyel megnöveljük, s.i.t. A KSS korábbi értéke által elfoglalt terület felszabadul, az Interpreter azt egyelőre nem használja fel.

- Ha a Karakterváltozó terület vége és a területen utoljára eltárolt string vége között nincs már akkora memóriarész, amely az új string tárolásához szükséges, akkor a PRIMO megkezd a Karakterváltozó területen elhagyott - már felszabadult - területek összegyűjtését. Ezt a tevékenységet szemétdűjtésnek - garbage collection - nevezik. Ennek során az Interpreter felderíti, hogy a Karakterváltozó területen hol található elhagyott mező, és ezeket - az értékes stringeket tároló területek kisebb memória címek irányába történő elmozdításával - a Karakterváltozó terület végén egyetlen összefüggő mezővé egyesíti. Így - miközben a karakteres változók Leíró-részében tárolt érték-cím mezők többségének tartalma az új helyzetnek megfelelően módosul - lehetővé válik, hogy az Interpreter lefoglalja az igényelt területet az új string számára. (Ha a Karakterváltozó terület tömörítése után sincs elegendő hely az új karaktersorozat tárolására, akkor a PRIMO OS - Out of String Space, Betelt a Karakterterület - hibát jelez.)

1.5. STACK bejegyzések

A PRIMO BASIC Interpretere a működéséhez szükséges rendszer-változók értékét a BASIC Interpreter változó területen őrzi. Néhány esetben azonban ez a megoldás nem alkalmazható, mert egyes rutinok működésük során szubrutinként önmagukat is meghívhatják (rekurzív hívás). Az újraindulás során a rutin felülírja RAM-ban lévő munkaváltozóit, így visszatérve a szubrutinból, az eredeti változó értékek már nem állnak rendelkezésre.

A fent leírt probléma megszüntetése érdekében a PRIMO azon rutinjai, amelyek megengedik az újraindítást - u.n. reentráns rutinok - munkaváltozóikat nem a BASIC Interpreter változó területen, hanem a STACK-ben őrzi.

Egy BASIC program végrehajtása során olyan esetben szükséges a rekurzív hívás, ha egy program-struktúra belsejében egy, a külső program-struktúrával azonos belső program-struktúra helyezkedik el. A BASIC programozási nyelvben három ilyen - különleges kezelést igénylő - struktúra fordulhat elő:

- egymásba ágyazott ciklusok (a ciklus magjában egy másik

ciklus van)

- egymásba ágyazott szubrutinok (a szubrutinban egy újabb szubrutin hívás szerepel)
- egymásba ágyazott kifejezések (egy kifejezésben - zárójelek közé zárva - egy rész-kifejezés található)

Ezekben az esetekben tehát a **BASIC** Interpreter bizonyos rutinokat egyszeresen - vagy akár többszörösen is - újraindít. A rekurzió okozta problémák megszüntetésére ilyenkor egy - a végrehajtott utasításra jellemző - adatcsomag kerül a **STACK**-be.

FOR-adatcsomag felépítése

Valahányszor a **BASIC** Interpreter a program végrehajtása során egy **FOR** utasítást észlel, meghatározza a végrehajtandó ciklus változójának memória címét, valamint kiértékeli a ciklus-változó kezdő- és végértékét, ill. a lépésközt megadó kifejezések aktuális értékét. Ezután beállítja a ciklusváltozó kezdőértékét, majd - felkészülve a ciklusok esetleges egymásba ágyazására - a ciklusváltozó végértékét, a növekményt, valamint néhány további információt egy adatcsomag formájában a **STACK**-be ment. E műveletek elvégzése után megkezdődik a ciklus magjának - amely tehát újabb **FOR** utasítást is tartalmazhat - végrehajtása.

A ciklusmag végét egy **NEXT** utasítás jelzi. Ekkor a **PRIMO BASIC** Interpreter megkeresi a **STACK**-ben azt az adatcsomagot, amely a most megtalált **NEXT** utasításhoz tartozó **FOR** utasítás végrehajtásakor került a **STACK**-be. (Az összetartozást a ciklus-változó címek összehasonlításával vizsgálja.) Ha ilyet nem talál, **NF - NEXT without FOR**, **FOR** nélküli **NEXT** - hibát jelez.

Ha megtalálja a keresett adatcsomagot, akkor kiemeli azt a **STACK**-ből, majd az abban talált adatoknak megfelelően megváltoztatja a ciklusváltozó értékét, és elvégzi a ciklus-vég vizsgálatot. Ha a ciklus magját ismételten végre kell hajtani, akkor az Interpreter visszairja a **STACK**-be az adatcsomagot, majd megkezd a ciklusmag 1. utasításának végrehajtását. Ha nem szükséges a ciklusmag ismételt végrehajtása, akkor a **BASIC** program következő utasításának értelmezését kezdi el.

Egy ciklus feldolgozása során végrehajtandó műveletek áttekintése után a következőkben ismertetjük a **FOR**-adatcsomag részletes felépítését. (Az egyes csomagok szavanként - 1 szó = 2 byte - kerülnek a **STACK**-be, ezért a leírásban **MSB**-vel jelöljük a szó értékesebb 8 bitjét, és **LSB**-vel a kevésbé értékes 8 bitet. Egy csomag 1. szava helyezkedik el a **STACK** tetején, vagyis a legkisebb memória címen. Az ezt követő szavak a **STACK** soronkövetkező szavaiban - magasabb címeken - állnak.)

----- A PRIMO SZÁMITÓGÉP MÓKODÉSE -----

FOR-adatcsomag egész típusú ciklusváltozó esetén

MSB	LSB
	FOR belső kódja
Ciklusváltozó RAM címe	
Ciklusváltozó típusa (= -1)	Lépésköz előjele
-----	-----
-----	-----
Lépésköz (egész típusú szám)	
Végérték (egész típusú szám)	
FOR utasítást tartalmazó sor sorszáma	
Ciklusmag 1. utasításának RAM címe	

FOR-adatcsomag egyszeres pontos valós ciklusváltozó esetén

MSB	LSB
	FOR belső kódja
Ciklusváltozó RAM címe	
Ciklusváltozó típusa (= 1)	Lépésköz előjele
- Lépésköz (egyszeres pontos valós szám) -	
- Végérték (egyszeres pontos valós szám) -	
FOR utasítást tartalmazó sor sorszáma	
Ciklusmag 1. utasításának RAM címe	

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

A bejegyzésekben látható "Lépésköz előjele" elnevezésű byte értéke rendre a következő:

Ha a lépésköz negatív	-	-1 (OFFH)
Ha a lépésköz pozitív	-	1
Ha a lépésköz = 0	-	0

Mint látható, a ciklusváltozó típusának függvényében két - részben különböző - adatcsomagot használ a PRIMO.

A FOR-adatcsomag ismeretében megérthető, hogy a PRIMO BASIC miért nem engedi meg, hogy a ciklusváltozó duplapontos valós, ill. indexelt változó legyen.

- Duplapontos valós ciklusváltozót azért nem alkalmazhatunk, mert ekkor a ciklusváltozó végértékét és a lépésközt is célszerűen duplapontos formában kellene tárolni, ami az ismertetett FOR-adatcsomagoktól eltérő - azoknál jelentősen nagyobb STACK-területet igénylő - bejegyzést igényelne.
- Indexelt változót ciklusváltozóként azért nem használhatunk, mert a FOR-adatcsomagban a PRIMO BASIC Interpretere eltárolja a ciklusváltozó RAM címét. Ha pedig a ciklusváltozó egy indexelt változó lenne, akkor a ciklus magjának végrehajtása alatt egy ezidáig még nem használt skalár változó előfordulása az indexelt változók táblájának memórián belüli eltolását eredményezné, de a FOR-adatcsomagban lévő ciklusváltozó-cím értéke még a régi RAM címre mutatna. Ez végső soron azt eredményezné, hogy a BASIC Interpreter nem találná meg az indexelt változók táblájában a ciklusváltozóhoz rendelt értéket.

GOSUB-adatcsomag felépítése

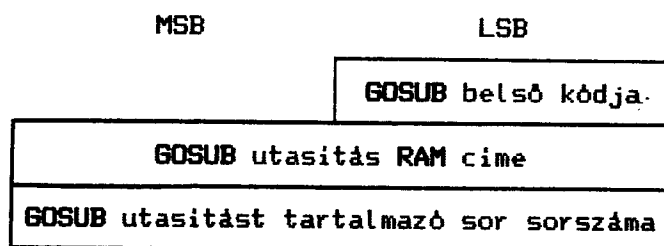
Amikor a PRIMO a BASIC program végrehajtása során felismer egy GOSUB utasítást, a STACK-ben egy GOSUB-adatcsomag formájában eltárolja a GOSUB utasítást tartalmazó BASIC programsor sorszámát - az aktuális sorszámot - és a GOSUB utasítás RAM címét. Ezután megkezdzi a kiválasztott szubrutin végrehajtását.

A szubrutin végét a RETURN utasítás jelzi. Ekkor a PRIMO BASIC Interpretere megkeresi a STACK-ben a GOSUB-adatcsomagot. Ha ilyet nem talál, RG - RETURN without GOSUB, GOSUB nélküli RETURN - hibát jelez. Ha megtalálja a bejegyzést, akkor az aktuális sorszámot a GOSUB-adatcsomagban talált sorszámmal teszi egyenlővé, majd az eltárolt RAM címtől kezdődően a GOSUB-ot követő BASIC utasítástól folytatja a program végrehajtását.

Egy szubrutin feldolgozása során végrehajtandó műveletek áttekintése után a következőkben ismertetjük a GOSUB-csomag

részletes felépítését. (Az egyes csomagok szavanként - 1 szó = 2 byte - kerülnek a **STACK**-be, ezért a leírásban **MSB**-vel jelöljük a szó értékesebb 8 bitjét, és **LSB**-vel a kevésbé értékes 8 bitet. Egy csomag 1. szava helyezkedik el a **STACK** tetején, vagyis a legkisebb memória címen. Az ezt követő szavak a **STACK** soronkövetkező szavaiban - magasabb címeken - állnak.)

GOSUB-adatcsomag



1.6. Kifejezések feldolgozása

A kifejezések feldolgozása a Kifejezés Kiértékelő - a továbbiakban **EXPRN** - alprogram feladata. Az **EXPRN** a **BASIC** program legkülönbözőbb helyein előforduló numerikus és/vagy karakteres kifejezéseket feldolgozza, majd eredményül egyetlen numerikus vagy karakteres értéket szolgáltat. Ennek során az **EXPRN** felbontja a zárójelezett kifejezéseket, és az abban található műveleteket megfelelő sorrendben végrehajtva állítja elő a kifejezés aktuális értékét. A kifejezés feldolgozása közben keletkező részeredményeket átmenetileg a **STACK**-ben tárolja. A végeredményt az **EXPRN** a Lebegőpontos Akkumulátorba helyezi.

Annak érdekében, hogy az ismertetett műveleteket az **EXPRN** végre tudja hajtani, végig kell olvasnia a teljes kifejezést. Ezt az adott kifejezés bal oldaláról indulva kezdi meg. Mindaddig halad előre a kifejezésben, amíg egy operátort nem talál, vagy el nem éri a **BASIC** utasítássor végét. Ha elért a sor végére, befejeződik a kifejezés feldolgozása. Ha egy operátort talált, akkor ezt az aktuális operátort és az operátor bal oldalán álló operandust - az u.n. aktuális operandust - összefüggő egységnek - u.n. csomagnak - tekinti, és vagy elhelyezi a csomagot a **STACK**-ben, vagy az aktuális operandus és a **STACK** tetején lévő csomagban lévő operandus között elvégzi a csomagban tárolt operációt, majd az eredményt és az aktuális operátort - mint egy új csomagot - visszahelyezi a **STACK**-be. Ezután az **EXPRN** folytatja a kifejezés további vizsgálatát.

Egy kifejezés feldolgozásának részletes menete a következő:

1. Az **EXPRN** beolvassa a kifejezés soronkövetkező első nem betűkód karakterét. Ennek típusától függően különféle műve-

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

leteket hajt végre.

- a. Ha a karakter számjegy vagy egy decimális pont, feldolgozza a soronkövetkező numerikus konstanst, és annak bináris értékét elhelyezi a lebegőpontos akkumulátorban.
- b. Ha a karakter egy idézőjel - " -, beolvassa a soronkövetkező karakteres konstanst, majd annak leíróját elhelyezi a Lebegőpontos Akkumulátorban.
- c. Ha a karakter egy betű, beolvassa a kifejezésben soronkövetkező - skalár vagy indexelt - változót, majd annak aktuális értékét elhelyezi a Lebegőpontos Akkumulátorban.
- d. Ha a karakter egy függvény tokene - belső kódja -, meghatározza a függvényértéket, majd elhelyezi azt a Lebegőpontos Akkumulátorban.
- e. Ha a karakter egy nyitózárójel - (-, feldolgozza a zárójelben álló kifejezést, majd annak aktuális értékét a Lebegőpontos Akkumulátorba tölti.
- f. Ha a karakter egy mínusz jel - negatív előjel -, feldolgozza a soronkövetkező kifejezést, és annak aktuális értékét betölti a Lebegőpontos Akkumulátorba.
- g. Ha a karakter a NOT logikai utasítás belső kódja, feldolgozza a soronkövetkező kifejezést, majd a kifejezés aktuális értéke egész részének 1-es komplementjét a lebegőpontos akkumulátorba tölti.
- h. Ha a karakter egy plusz jel - pozitív előjel -, azt figyelmen kívül hagyja, mivel a pozitív előjel megadása, vagy elhagyása nem változtatja meg a kifejezés értékét.
- i. Ha nincs már több karakter - elérte a BASIC utasítássor végét -, MO - Missing Operand, Hiányzó operandus - hibát jelez.

(A d,e,f és g esetben a soronkövetkező kifejezés feldolgozására újra indul az EXPRN alrutin - rekurzív hívás -.)

2. Beolvassa az operandust követő karaktert. Ha ez egy olyan karakter - vagy belső kód - amely kifejezésben nem szerepelhet, akkor ezt az EXPRN úgy értelmezi, hogy elérte a kifejezés végét. Ha ez egy operátor belső kódja, akkor azt a

továbbiakban az **EXPRN** az aktuális operátornak tekinti. Ezután a további feldolgozás a 3. ponttól folytatódik.

3. Ha az **EXPRN** elért a kifejezés végére, akkor a 3/b. pont alatt leírt tevékenységeket hajtja végre. Ha nincs még a kifejezés végén, akkor az aktuális operátor típusától függően a következő tevékenységek valamelyikét hajtja végre:
 - a. Amennyiben az aktuális - az éppen megtalált - operátor az operátorok végrehajtási sorrendjében magasabb szinten áll - vagyis nagyobb a precedenciája (ld. 1. RÉSZ 1.7.6.) - mint a **STACK** tetején lévő csomag operátora, akkor az **EXPRN** az aktuális operandust és az aktuális operátort - vagyis a csomagot - elhelyezi a **STACK**-ben, majd a kifejezés további feldolgozását az 1. ponttól folytatja. (Egy kifejezés feldolgozásának megkezdésekor a feltételezett precedencia értéke = 0, így bármely előforduló operátor esetén a csomag a **STACK**-be kerül.)
 - b. Ha az aktuális operátor végrehajtási szintje - precedenciája - azonos vagy kisebb, mint a **STACK** tetején lévő csomag operátoráé, ill. az **EXPRN** elérte a kifejezés végét, akkor az alprogram kiemeli a **STACK**-ból a legfelső csomagot, és végrehajtja a csomagban tárolt operátor által kijelölt műveletet. (A kiemelése során a **PRIMO** először az adott csomagot értelmező rutin **ROM**-címét olvassa ki a **STACK**-ból. A további adatok kiolvasását és értelmezését már a csomagokhoz rendelt különböző rutinok végzik.) A művelet 1. operandusa a csomagban tárolt érték, 2. operandusa az aktuális operandus értéke lesz. A keletkező eredményt az **EXPRN** a továbbiakban aktuális operandusnak tekinti és a Lebegőpontos Akkumulátorban tárolja. (Ha az 1. és 2. operandus eltérő típusú, az **EXPRN** automatikusan végrehajtja a szükséges típuskonverziót. Ennek során mindig a kevesebb byte-on ábrázolt operandust alakítja át a másik operandus típusára.) Ezután a kifejezés kiértékelése a 3. ponttól folytatódik. Amennyiben az **EXPRN** a **STACK**-ben nem talál további kiemelhető csomagot, befejeződik a kifejezés kiértékelése, az **EXPRN** visszatér a hívó programhoz. Az aktuális numerikus érték, vagy a karakteres érték leírja a Lebegőpontos Akkumulátorban található.

Az **EXPRN** működésének áttekintése után a következő táblázatban felsoroljuk a kifejezésekben szerepeltethető operátorokat, valamint az **EXPRN** által az egyes operátorokhoz rendelt belső kódokat és a decimális, ill. hexadecimális precedencia értékeket.

----- A PRIMO SZÁMÍTÓGÉP MŰKÖDÉSE -----

Operátor	Belső kód	Precedencia kód	Megjegyzés
↑	---	127;7F	A hatványozás műveletét az EXPRN az aritmetikai műveletektől elkülönítve végzi el, ezért nincs belső kódja.
*	2	124;7C	
/	3	124;7C	
+	0	121;79	
-	1	121;79	125;7D ha unáris
)	1	100;64	A logikai operátorokat az EXPRN az aritmetikai operátoroktól elkülönítve dolgozza fel, így azok belső kódja megegyezhet az aritmetikai operátorok belső kódjával.
=	2	100;64	
)= =)	3	100;64	
(	4	100;64	
())(	5	100;64	
(= =(	6	100;64	
NOT	---	90;5A	Az elkülönített végrehajtás miatt nincs belső kód.
AND	---	80;50	
OR	---	70;46	

Amint már említettük, az **EXPRN** működése során a kifejezés kiértékelés közbülső eredményeit - a csomagokat - a **STACK**-ben tárolja. E csomagok felépítése a bennük eltárolt operandusok típusától - egész típusú, egyszerespontos vagy duplapontos valós, karakteres - és az eltárolt operátortól függően különböző. A következőkben ismertetjük valamennyi lehetséges csomag felépítését. (Az egyes csomagok szavanként - 1 szó = 2 byte - kerülnek a **STACK**-be, ezért a leírásban **MSB**-vel jelöljük a szó értékesebb 8 bitjét, és **LSB**-vel a kevésbé értékes 8 bitet. Egy csomag 1. szava helyezkedik el a **STACK** tetején, vagyis a legkisebb memória címen. Az ezt követő szavak a **STACK** soronkövetkező szavaiban - magasabb címeken - állnak. A csomagokban lévő "Folytatás ROM címe" elnevezésű szó az **EXPRN** alprogram azon részének ROM címét tárolja, amely a kifejezés feldolgozásának 1. pontban leírt tevékenységét hajtja végre.)

"Aritmetikai operátor" csomag

MSB	LSB
Előző operátor precedenciája	-----
Folytatás ROM címe	
Egész típusú, egyszeres pontos vagy duplapontos valós adat (1,2,4*2 byte)	
Adat típusa	Operátor kódja
Aritmetikai műveletvégző rutin ROM címe	

"Hatványozás" csomag

MSB	LSB
Előző operátor precedenciája	-----
Folytatás ROM címe	
Egyszeres pontos valós adat (2*2 byte)	
Hatványozást végző rutin ROM címe	

"Logikai operátor" csomag

MSB	LSB
Előző operátor precedenciája	-----
Folytatás ROM címe	
Egész típusú adat (1*2 byte)	
Logikai műveletvégző rutin ROM címe	

----- A PRIMO SZAMITOGEP MOKODESE -----

"Reláció karakteres adattal" csomag

MSB	LSB
Előző operátor precedenciája	-----
Relációs operátor belső kódja	-----
Folytatás ROM címe	
Karakter leíró címe	
Karakteres reláció rutin ROM címe	

"Reláció numerikus adattal" csomag

MSB	LSB
Előző operátor precedenciája	-----
Relációs operátor belső kódja	-----
Folytatás ROM címe	
Egész típusú, egyszerespontos vagy duplapontos valós adat (1,2,4*2 byte)	
Adat típusa	Operátor kódja
Numerikus reláció rutin ROM címe	

1.7. Területfoglalás gépi kódú rutinok számára

A PRIMO lehetővé teszi, hogy a Felhasználó BASIC programjából saját gépi kódú rutinokat indítson (ld. CALL függvény). Ez a lehetőség biztonságosan azonban csak akkor alkalmazható, ha a gépi rutinok számára a memóriában olyan területet tudunk biztosítani, amelyet a BASIC program működése során - ellenőrizetlenül - nem tud felülni. A következőkben bemutatjuk, hogy a PRIMO-ban hogyan lehet ilyen területeket létrehozni.

Gépi kódú programunkat elvileg három különböző területen helyezhetjük el. A következőkben ismertetjük a különféle megoldások előnyeit, ill. hátrányait.

a. **BASIC változóban.**

E megoldás előnye, hogy a területfoglalás a **BASIC** program futása alatt hajtható végre. Ha skalár változót használunk, akkor egy egész típusú változóban 2, egyszerespontos valós változóban 4, duplapontos valós változóban 8 byte hosszúságú gépi kódot helyezhetünk el. (Karakteres változót gépi kód tárolására ne használjunk.) Ebből látható, hogy skalár változók alkalmazása esetén csak igen kisméretű gépi program futtatható. Ezen a problémán úgy segíthetünk, hogy a gépi kódokat egy megfelelő méretre dimenzionált vektorban helyezzük el. (A vektor tetszőleges numerikus típusú lehet, karakter-vektort gépi kód tárolására ne használjunk.) Ekkor arra kell ügyelnünk, hogy a vektor memóriacímének meghatározását végző **VARPTR** függvény alkalmazása után a programban korábban még nem használt skalár változó előfordulása a tömbváltozók címét - így a gépi kódot őrző vektor címét is - megváltoztatja. Ezt a problémát úgy szüntethetjük meg, hogy a **BASIC** programunk elején egy fiktív értékadásban feltüntetjük mindazon skalár változókat, amelyeket a **BASIC** programban használni kívánunk.

Ha gépi kódunkat **BASIC** változóban tároljuk, feltétlenül figyelembe kell vennünk azt, hogy a **BASIC** program módosításának hatására - törlés, beszúrás - a kódokat tároló változó memóriacíme módosul. Ezért így csak olyan gépi kódokat helyezhetünk el, amelyekben nincs abszolút cím. (Ez azt jelenti, hogy a gépi kódú programban csak relatív vezérlésátadó utasításokat alkalmazhatunk, ha a vezérlésátadás a saját rutinunk egy kiválasztott pontjára irányul. Természetesen szerepelhet a gépi kódban abszolút cím is, ha az pl. egy **ROM**-ban elhelyezkedő szubrutin címe, vagy valamelyik rendszerváltozó **RAM** címe. Megvalósítható az is, hogy a gépi kódú program relokálását maga a **BASIC** program hajtsa végre.)

b. **A magnó puffer és a BASIC programterület között.**

Ha a gépi kódú programot a magnó puffer és a **BASIC** programterület között kívánjuk elhelyezni (ld. memória felosztás, F-27 oldal), akkor tudomásul kell vennünk, hogy ezt az elhelyezendő gépi rutinokat felhasználó **BASIC** program nem tudja megtenni. Ezért ez a megoldás csak úgy alkalmazható, hogy a területfoglaláshoz szükséges utasításokat parancs - direkt - módban adjuk meg, vagy egy a területfoglalást végrehajtó **BASIC** programot futtatunk, majd a szükséges terület biztosítása után a **PRIMO** memóriájába betöltött **BASIC**, vagy gépi kódú program már elhelyezheti rutinjait a lefoglalt területen.

E megoldás előnye viszont az, hogy a lefoglalt memória terület kezdőcíme mindig a 17386;43EA lesz, így abszolút

----- A PRIMO SZAMITOGEP MOKODESE -----

cimeket, pl. saját szubrutinokat tartalmazó gépi kódú programot is alkalmazhatunk.

Az igényelt terület lefoglalását végrehajtó **BASIC** utasítás-sorozat - részletesebb magyarázat nélkül - a következő:

```
POKE 16548,17386+X-256*INT((17386+X)/256),INT((17386+X)/256)
POKE 17385+X,0
NEW
```

Az utasításokban szereplő **X** az igényelt terület byte-okban megadott méretét jelöli.

Annak érdekében, hogy a területfoglalást a **BASIC** Interpreter tudomásul vegye, a **NEW** parancsot kell alkalmazni. Ez teszi lehetetlenné, hogy egy **BASIC** program saját maga számára a memória elején területet biztosítson, hiszen a szükséges műveletek elvégzése után - az érdemi tevékenység megkezdése előtt - kitörli magát a **PRIMO** memóriájából.

c. A karakterváltozó terület és a Képernyő RAM között.

Ha a gépi kódú programok/változók számára a Képernyő RAM előtt akarunk memória területet foglalni, akkor azt a futó **BASIC** program segítségével is megtehetjük. Így pl. lehetséges, hogy egy magnóról betöltött program, elindítása után gépi rutinjainak maga foglalja le a szükséges területet. Ezután már a megfelelő kódok **POKE** utasításokkal az elkülönített területre elhelyezhetők.

Mivel a különféle memória méretű - A-32, A-48, A-64 - gépekben a Képernyő RAM terület kezdőcíme más és más, ezért a lefoglalt terület kezdőcíme is függeni fog azon **PRIMO** típusától, amelyben a területfoglalást végrehajtjuk. Ezért az elkülönített területre elhelyezendő gépi kód - hasonlóan az a. pontban leírtakhoz - csak megkötésekkel tartalmazhat abszolút memória cimeket.

A területfoglalást a következő **BASIC** utasításokkal valósíthatjuk meg: (A programrészletben szereplő **X** az igényelt terület byte-okban megadott méretét jelöli.)

```
100 CIM=256*PEEK(16562)+PEEK(16561)-X
110 POKE 16561,CIM-256*INT(CIM/256),INT(CIM/256)
120 CLEAR 50
130 CIM=256*PEEK(16562)+PEEK(16561)
```

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

Az előbbi programrészletet célszerű a BASIC program elejére írni, mivel a 120-as címkéjű sorban szereplő CLEAR utasítás valamennyi változó értékét törli. Ugyancsak a CLEAR utasítás miatt a fenti programrészlet nem lehet egy ciklus magjában, ill. szubrutinban (ld. 1. RÉSZ 2.2. pontban a CLEAR utasítás részletes ismertetését.). Ha a BASIC programban a karakteres változók számára több, mint 50 byte szükséges, akkor természetesen a CLEAR operandusa értelemszerűen megváltoztatható. A BASIC program további részében a lefoglalt terület kezdőcímét a CIM nevű változó tárolja. Az ismertetett okokból a különféle típusú - A-32, A-48, A-64 - PRIMO-knál a CIM változó értéke különböző lesz. Hogy ezt elkerüljük, a programrészlet 100-as sorszámu sorát - a rendelkezésre álló memória minimális méretének figyelembe vételével - megváltoztathatjuk. Ha pl. a

100 CIM=25624

BASIC sort alkalmazzuk, akkor a lefoglalt terület kezdőcíme 25624 lesz, míg mérete A-32 típus esetén 1000, A-48-asnál 17384, A-64-esnél 33768 byte lesz.

Annak érdekében, hogy egy magnóról betöltött és elindított program egyszerűen meg tudja határozni a kiépített memória méretét, a ROM 20-as című byte-jának értéke eltérő a különböző típusú PRIMO-knál. Az említett cím tartalma rendre a következő:

PRIMO	A-32	-	104
	A-48	-	168
	A-64	-	232

----- FELJEGYZÉSEK -----

2. FEJEZET

A PRIMO aritmetikai rutinjai

Ebben a fejezetben összefoglaljuk a PRIMO aritmetika rutinjainak felhasználásához szükséges alapvető ismereteket. Bemutatjuk az alkalmazott számbábrázolási módokat, megadjuk az aritmetikai rutinok munkaregisztereinek pontos értelmezését. Ismertetjük az alpműveleteket és függvényeket végrehajtó alprogramok belépési címeit, paramétereit. (A megadott RAM- és ROM-címek a PRIMO '84.1 verziójára vonatkoznak. Az értékek a PRIMO továbbfejlesztésekor megváltozhatnak.)

2.1. A PRIMO számbábrázolási formátumai

A PRIMO aritmetikai programcsomagja két, alapvetően különböző típusú számbábrázolással rendelkezik. A következőkben e két ábrázolási mód részletes leírásán túlmenően konkrét példákon keresztül igyekszünk bemutatni a számok konverzióját és ábrázolását.

Egész típusú számbábrázolás

Az egész típusú számbábrázolás a fixpontos ábrázolási mód speciális esete. Ennél az ábrázolási módnál ugyanis a tizedespont - vagy bináris pont - a szám ábrázolására lefoglalt terület jobb szélén helyezkedik el. Ebből következik, hogy az ily módon ábrázolt számok törtrészt nem tartalmazhatnak. Az ábrázolási tartományt, vagyis a pozitív és negatív tartományban ábrázolható maximális érték nagyságát a szám tárolására felhasznált terület hossza - azaz a bitek száma - határozza meg.

A PRIMO esetében az egész típusú szám tárolására két Byte szolgál. Mivel az ábrázolás úgynevezett kettes komplementes kódú, ezért a számbábrázolás értéktartománya:

$$-32768 \dots +32767$$

Az egész típusú számokat a PRIMO a következő formában ábrázolja:

2^{15}	2^{14}	2^{13}	$\dots$	2^2	2^1	2^0
----------	----------	----------	---------	-------	-------	-------

↑
Előjelet jelképező bit

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

A kettes komplement kódú ábrázolás azt jelenti, hogy pozitív szám esetén a szám abszolút értékét kifejező bináris számjegyek úgy töltik ki a bitpozíciókat, mintha a képzeletbeli bináris pont az ábrázolásra fenntartott bitpozíciók után következne. A bináris szám értékes jegyeit megelőző összes pozícióba nulla kerül. Így automatikusan a 15. - az előjelet ábrázoló - pozíción is nulla fog szerepelni, amely ennél az ábrázolási módnál megegyezés szerint a pozitív előjelet jelenti. A legnagyobb ábrázolható pozitív szám ezek szerint:

$$0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 = 2^{15}-1 = 32767$$

Negatív szám tárolása esetén első lépésként a szám abszolút értékét, mint egy pozitív számot az előbb leírtak szerint ábrázoljuk. Ezután a pozitív számot bitenként invertáljuk, vagyis minden pozícióban az ott lévő bit értékét az ellenkezőjére változtatjuk. Legvégül a részeredményhez a legalacsonyabb helyiértéken 1-et hozzáadunk, és így megkapjuk az eredeti szám úgynevezett kettes komplement kódú megfelelőjét. Ha az ábrázolást helyesen végezzük, akkor a legmagasabb helyiértéken - a 15. pozíción - egy 1 értékű bit jelzi az ábrázolt egész típusú szám negatív előjelét. Ezt a leírt algoritmus minden esetben biztosítja, tehát külön nem kell az előjel beállításával törődni. A legnagyobb ábrázolható negatív szám:

$$1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 = -2^{15} = -32768$$

Ha egy kettes komplement kódban ábrázolt számot értelmezni akarunk, akkor a bináris alak 15. bitpozíciójában lévő érték egyrészt megadja a szám előjelét, másrészt megszabja a visszaalakítás módját is. Amennyiben az előjel pozícióban nulla szerepel, akkor a szám pozitív, és a visszaalakítás során a pozíciókhoz rendelt helyiértékek által jelképezett 2 hatványok összeadásával kell a szám decimális értékét meghatározni. Ha a legmagasabb helyiértéken lévő bit értéke 1, akkor a decimális konverzió teljesen hasonló módon történik mint az ábrázolás, vagyis először invertálni kell minden egyes pozíció tartalmát, majd a részeredményhez 1-et hozzá kell adni. Az így kapott szám már pozitív számként - az ott leírtak alapján - alakítható vissza.

Az egész típusú, előjeles számok kettes komplement kódban történő ábrázolásának előnye, hogy így minden művelet visszavezethető az előjelbittel együtt elvégzett összeadásra, illetve kivonásra. Aritmetikai műveletvégzés - összeadás, kivonás - közben az előjelbit nem igényel külön feldolgozást, ugyanúgy kell kezelni mint az utána következő 15 bitet. A műveletvégzés során az előjelbit automatikusan a helyes értékre áll be. Csak az eredmény értelmezésekor kell a legértékesebb bitpozíción álló előjelbitet külön kezelni.

Példák az egész típusú számbábrázolásra:

1. Ábrázoljuk a $+2511_{10}$ -et.

$$2511_{10} = 100111001111_2$$

Az átalakított számot a számbábrázolási hosszra - 16 bit - kiegészítve az eredmény:

0 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1

2. Ábrázoljuk a -7936_{10} -at.

$$7936_{10} = 1111100000000_2$$

Az ábrázolási hosszra kiegészítve:

0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0

A bitenkénti invertálást elvégezve:

1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1

A részeredményhez 1-et hozzáadva megkapjuk a helyes eredményt:

1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0

3. Alakítsuk vissza decimálissá a következő, egész típusú számbábrázolási móddal ábrázolt bináris számot.

0 0 0 1 1 0 1 0 1 1 1 0 1 0 1 0

Mivel a 15. pozíción lévő előjelbit nulla, ezért - lévén pozitív szám - közvetlenül felírhatjuk a számot a megfelelő helyiértékekhez tartozó 2 hatványok összegeként:

$$2^{12} + 2^{11} + 2^9 + 2^7 + 2^6 + 2^5 + 2^3 + 2^1 = 6890_{10}$$

4. Alakítsunk vissza decimális számmá egy másik, kettes komplementes kódban ábrázolt számot.

1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0

Mivel az előjelbit 1, ezért a szám negatív lesz.

Első lépésben végezzük el a bitenkénti invertálást.

0 0 0 0 0 1 0 1 0 1 0 1 1 1 0 1

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

Adjunk hozzá a számhoz 1-et.

0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 0

Irjuk fel a számot az egyes helyiértékekhez tartozó 2 hatványok összegeként:

$$2^{10} + 2^8 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 = 1374_{10}$$

Az előjel figyelembevételével a helyes eredmény: -1374

Valós típusú számábrázolás

Az egész típusú tárolási mód két jelentős hátrányát - az aránylag szűk ábrázolási tartományt és a törtrész hiányát - a valós típusú számábrázolás küszöböli ki. A valós típusú tárolási mód lehetőséget biztosít bármilyen törtszám bizonyos pontossággal történő ábrázolására.

Az ábrázolás alapgondolata, hogy bármely szám felírható u.n. exponenciális alakban, azaz egy együttható és egy hatvány szorzataként.

$$A = M \cdot 10^K$$

ahol: A - tetszés szerinti szám

M - együttható u.n. Mantissza

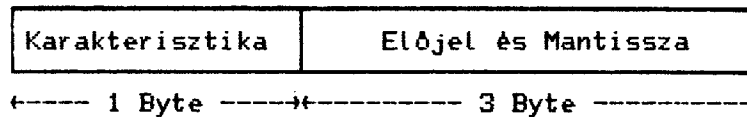
K - a hatvány Kitevője, u.n. Karakterisztika

Az így felírt számot normál alakra hozva - vagyis úgy átalakítva, hogy a Mantissza csak törtrészt tartalmaz és a törtrész első nullától különböző számjegye közvetlenül a tizedespont mögött áll - u.n. normalizált számot kapunk. Ebből a felírási módból nyilvánvalóan adódik, hogy valós számábrázolási módban két értéket kell tárolni, a Mantisszát és a Karakterisztikát. A hatvány alapját, azaz a számrendszer alapszámát nem kell tárolni, hiszen az konstans. Mivel a számítógép bináris ábrázolási rendszeréhez közelebb állnak kettő hatványai mint a tízé, ezért a számrendszer, és így az ábrázolás alapjául általában kettőt vagy tizenhatot szoktak választani.

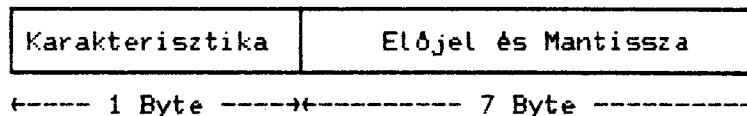
A valós típusú számábrázolásnál egy szám tárolására szolgáló területet meg kell osztani a Mantissza és a Karakterisztika között. A Mantisszabitek számának növelésével az ábrázolás pontossága növelhető, a Karakterisztika bitszámának növelése a legnagyobb ábrázolható szám értékének növelését eredményezi. A PRIMO esetében két különböző valós típusú számábrázolásról beszélhetünk. E két különböző forma csupán a Mantisszabitek számában tér el. A különbség az ábrázolás pontosságában jelentkezik. Az egyszerűsített mód esetében a Mantissza tárolására 24 bit szolgál,

ezzel 7 decimális számjegy pontosság érhető el. Duplapontos módban a PRIMO 56 biten tárolja a Mantisszát. Ez 17 decimális számjegy pontosságot eredményez. A Karakterisztika, ill. a Mantissza előjelének ábrázolása mindkét mód esetén azonos.

Valós típusú egyszeres pontos ábrázolási forma



Valós típusú duplapontos ábrázolási forma



A Mantissza ábrázolása

A Mantissza tárolására 24, ill. 56 bit áll rendelkezésre. A szám bináris formára történő átalakítása után a PRIMO a részeredményt 2-es alapon normalizálja. Ennek során a bináris pontot a Mantissza balról haladva első 1-et tartalmazó bitje elé helyezi. Ezzel egy időben - a bináris pont egy bittel történő elmozgatásakor - a Karakterisztika értéke módosul. A bináris pont jobbra mozdításakor a Karakterisztika értéke eggyel csökken, balra léptetéskor a Karakterisztika értéke eggyel növekszik. Az így kapott normalizált szám a Mantissza tárolására fenntartott terület legértékesebb helyiértékén biztosan 1-et tartalmaz. Mivel egy lebegőpontos számot a PRIMO mindig normalizáltan tárol, ezért a Mantissza legértékesebb bitjét - ami mindig 1-et tartalmazna - a tárolt szám előjelének ábrázolására használjuk fel. Természetesen a szám feldolgozásakor figyelembe kell venni, hogy a legértékesebb Mantissza biten az előjel áll. Így az ábrázolási pontosság - változatlan Mantissza bitszám mellett is növelhető. Megegyezés szerint a legértékesebb Mantissza bit 0-ás értéke pozitív számot, 1-es értéke negatív számot jelképez.

A Mantissza értékének meghatározásakor - a konverzió során - a legkisebb ábrázolási hiba érdekében a PRIMO esetenként a számított Mantissza értéket kerekítve tárolja.

A Karakterisztika ábrázolása

A normalizálás során kapott Karakterisztika érték u.n. alapértékes ábrázolási módon kerül tárolásra. Szokás ezt a módszert 128 többletes Kitevő ábrázolásnak is nevezni. E módszer lényege, hogy a Kitevő valódi értékét az alapértékkel módosítva tároljuk.

----- A PRIMO SZAMITOGEP MOKODESE -----

A 8 biten ábrázolható 0...255 különféle lehetséges számértéket felező 128. állapotot önkényesen 0-nak - alapértéknek - választjuk. Tároláskor a Kitevő előjeles valódi értékét az alapértékhez hozzáadva kapjuk a módosított Kitevőt. A Karakterisztika előjelét pozitívnak tekintjük, ha értéke nagyobb vagy egyenlő, mint 128 és negatívnak, ha a Karakterisztika értéke 128-nál kisebb. Ezzel a módszerrel

$$2^{-127} \dots 2^{+127}, \text{ azaz } 10^{-38} \dots 10^{+38}$$

nagyságrendű számok ábrázolhatók. Ha a tárolandó szám 10^{-38} -nál kisebb, akkor a PRIMO - a Mantissza értékét figyelmen kívül hagyva - Karakterisztikaként 0-át tárol, vagyis a Karakterisztikájában 0 értéket hordozó számot u.n. gépi nullának tekinti.

Az egyszeres pontos és duplapontos valós számaábrázolás között csupán a Mantissza hosszában - ezzel az ábrázolási pontosságban - van különbség. A Karakterisztika képzése, tárolása, és így az ábrázolható számok nagyságrendje mindkét módnál megegyezik.

Példák valós típusú számaábrázolásra

1. Ábrázoljuk egyszeres pontos valós módban a következő számot:

$$753.17_{10}$$

$$753_{10} = 1011110001_2$$

$$0.17_{10} = 0.00101011100001_2$$

$$753.17_{10} = 1011110001.00101011100001_2$$

normalizálás után az eredmény:

$$0.101111000100101011100001_2 \cdot 2^{10}$$

az előjel beállítása után a Mantissza végleges értéke:

$$001111000100101011100001$$

a Karakterisztika értéke:

$$128+10 = 138_{10} = 10001010_2$$

a teljes eredmény:

$$10001010 \ 00111100 \ 01001010 \ 11100001$$

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

2. Ábrázoljuk egyszerespontos valós módban a következő számot:

-19728.65

$$19728_{10} = 100110100010000_2$$

$$0.65_{10} = 0.1010011001_2$$

$$19728.65_{10} = 1000110100010000.1010011001_2$$

normalizálás után az eredmény:

$$0.1001101000100001010011001_2 * 2^{15}$$

Az előjel negatív, ezért a Mantissza legértékesebb bitpozíciójában álló érték 1 marad. Mivel a mantisszát 24 biten ábrázoljuk és az első elmaradó bit értéke 1, ezért a mantisszát felfelé kerekítjük. A Mantissza végleges értéke:

100110100010000101001101

a Karakterisztika értéke:

$$128+15 = 143_{10} = 10001111_2$$

a teljes eredmény:

10001111 10011010 00100001 01001101

3. Ábrázoljuk duplapontos valós módban a következő számot:

$$0.005715_{10}$$

$$0.005715_{10} = 0.0000000101110110100010011100101000011000101111010110011000100111_2$$

normalizálás után az eredmény:

$$0.1011101101000100111001010000110010101111010110011000100111_2 * 2^{-7}$$

az előjel beállítása és a szükséges kerekítés végrehajtása után a Mantissza végleges értéke:

00111011 01000100 11100101 00001100 01011110
10110011 00010100

a Karakterisztika értéke:

$$128+(-7) = 121_{10} = 01111001_2$$

----- A PRIMO SZÁMITÓGÉP MÓKODÉSE -----

a teljes eredmény:

01111001 00111011 01000100 11100101
00001100 01011110 10110011 00010100

4. Ábrázoljuk duplapontos valós módban a következő számot:

0.1

$$0.1_{10} = 0.0001100110011001100... \text{ (végtelen hosszú)}$$

normalizálás után az eredmény:

$$0.110011001100..._2 * 2^{-3}$$

az előjel beállítása és a szükséges kerekítés végrehajtása után a Mantissza végleges értéke:

01001100 11001100 11001100 11001100 11001100
11001100 11001101

a Karakterisztika értéke:

$$128 + (-3) = 125_{10} = 01111101_2$$

a teljes eredmény:

01111101 01001100 11001100 11001100
11001100 11001100 11001100 11001101

2.2. Az aritmetika munkaregiszterei

A következőkben ismertetjük az aritmetika munkaregisztereit. Megadjuk az egyes változók értelmezését, valamint decimális és hexadecimális címeiket.

Lebegőpontos Akku. #1	Lebegőpontos Akku. #2	Egész	Egyszeres	Dupla
16669;411D	16679;4127	---	---	LSB
16670;411E	16680;4128	---	---	NMSB
16671;411F	16681;4129	---	---	NMSB
16672;4120	16682;412A	---	---	NMSB
16673;4121	16683;412B	LSB	LSB	NMSB
16674;4122	16684;412C	MSB	NMSB	NMSB
16675;4123	16685;412D	---	MSB	MSB
16676;4124	16686;412E	---	EXP	EXP

EXP - Karakterisztika
 MSB - Mantissza legértékesebb bitjét tartalmazó Byte
 NMSB - Mantissza soronkövetkező Byte
 LSB - Mantissza legkevésbé értékes bitjét tartalmazó Byte

A fenti táblázat az aritmetika két legfontosabb munkaváltozójának memórián belüli elhelyezkedését mutatja. Az aritmetika által végrehajtott műveletek különböző típusú - egész, egyszeres-pontos, ill. duplapontos valós - eredménye a Lebegőpontos Akkumulátor #1-ben keletkezik. A keletkezett eredmény típusára jellemző kód a 16559;40AF című változóban található. A típusok és az azokat azonosító kódok a következők:

egész típusú érték	= 2
karakteres érték	= 3
egyszeres pontos érték	= 4
duplapontos érték	= 8

Az aritmetika a Lebegőpontos → Karakteres konverzió számára rendelkezik egy 26 Byte méretű munkaterülettel. A terület kezdőcíme 16688;4130. Ez az a terület, ahová a konvertáló rutin az átalakított bináris érték karakteres megfelelőjét - a számot jelképező karaktersorozat - helyezi el.

2.3. A Karakteres ↔ Bináris konverzió rutinjai

A következőkben ismertetjük a PRIMO aritmetika azon rutinjait, amelyek a 2.1. fejezetben leírt Karakteres decimális → Bináris, ill. Bináris → Karakteres decimális konverziókat elvégzik.

Karakteres decimális értéknek a memóriában elhelyezkedő azon karaktersorozat - stringet - nevezzük, amely egy egész, tört vagy exponenciális alakban megadott decimális szám előjelét, számjegyeit, tizedes pontját és Kitevőjét tartalmazza.

Karakteres decimális → Bináris konverziós rutinok

A konverziós rutinok a HL regiszterpár tartalma által meg-címzett memória területen elhelyezkedő stringet decimális számnak értelmezik, és eredményül - különféle számbázisú módban - annak Bináris megfelelőjét szolgáltatják.

Karakteres → Egész típusú konverzió Belépési cím: 07770;1E5A

Egész típusú konverziós rutin. Az eredményként keletkező egész típusú érték a rutinból való visszatéréskor a DE regiszterpárban található. HL az első karakter címét kell, hogy tartalmazza. Előjel nélküli konverziót végez. 0-65529-ig!

----- A PRIMO SZAMITOGEP MOKODESE -----

Karakteres → Bináris konverzió

Belépési cím: 03692;0E6D

Ha az átalakítandó string nem tartalmaz tizedespontot, ill. Kitevőt, és a keletkező szám értéke a -32768...32767 intervallumba esik, akkor eredményül egy egész típusú számot kapunk. Amennyiben az előbbi feltételek nem teljesülnek, akkor az eredmény egyszerespontos valós szám lesz. Ha a karaktersorozatban álló szám egészrészében lévő számjegyek száma több mint 7, ill. a Kitevőben duplapontos Karakterisztika - D - áll, akkor duplapontos valós szám keletkezik. Az eredmény minden esetben a Lebegőpontos Akkumulátor #1-ben található. A keletkezett eredmény típusa a 16559;40AF címen lévő munkaváltozóban található.

Karakteres → Duplapontos konverzió

Belépési cím: 03685;0E65

Duplapontos valós konverziós rutin. Az eredményként keletkező duplapontos valós érték a rutinból történő visszatéréskor a Lebegőpontos Akkumulátor #1-ben található.

Bináris → Karakteres decimális konverziós rutinok

Az alábbi rutinok az eredményül szolgáltatott Karakteres decimális értéket a 16688;4130 címen kezdődő munkaterületre, ill. az Egész típusú → Karakteres konverziós rutin esetében a híváskori HL regiszterpár által megcímezett területre tárolják.

Egész típusú → Karakteres konverzió

Belépési cím: 04911;132F

A Lebegőpontos Akkumulátor #1-ben tárolt számértéket a HL regiszterpár által kijelölt memóriacímre konvertálja.

Bináris → Karakteres konverzió

Belépési cím: 04029;0FBD

A Lebegőpontos Akkumulátor #1-ben tárolt, tetszőleges típusú bináris számot Karakteres decimális értékke alakítja, majd az eredmény stringet a 16688;4130 címen kezdődő munkaterületre tárolja. #1 elvesszik

Bináris → Karakteres konverzió formátum megadással

Belépési cím: 04030;0FBE

Működése megegyezik az előbbieken leírt rutinnal, de lehetőség van a keletkező eredmény string formátumának meghatározására. A formátumot meghatározó paramétereket a rutin meghívásakor az A, B és C regiszterekben kell megadni. Az egyes regiszterek jelentése a következő:

A regiszter:

Bit7 = 0, Nincs szükség a formázásra
 Bit6 = 1, Minden, a tizedes ponttól balra haladva 3. szám-
 jegy előtt álljon egy vessző
 Bit5 = 1, A szám előtti kihasználatlan számjegypozíciókon *
 karakterek álljanak
 Bit4 = 1, A szám elején jelenjék meg egy \$ karakter
 Bit3 = 1, Jelenjen meg a pozitív előjel is
 Bit2 = 1, az előjel a szám után jelenjen meg
 Bit1 = Fenntartva
 Bit0 = 1, A szám féllogaritmikus alakban jelenjen meg

B regiszter:

Értéke a decimális pont bal oldalán álló számjegyek számát
 határozza meg. az előjelet is bele kell kalkulálni

C regiszter:

Értéke a decimális pontot követő számjegyek számát határozza
 meg. Ha a szám a megadott értéknél több tizedesjegyet tar-
 talmaz, a PRIMO a számot a megadott számjegyre kerekíti.

- 1 2 számjegyre 3
 kell!

2.4. Aritmetikai műveletek rutinjai

Ezen fejezetben megadjuk az aritmetika egyes rutinjainak felhasználásához szükséges paraméterek értelmezését. A rutinok kezdőcímének decimális és hexadécimális értékén kívül ismertetjük az egyes alprogramok meghívásakor átadandó információ, ill. a rutin visszatérése után szolgáltatott értékek értelmezését. A következő fejezetekben a különböző számaábrázolási módokhoz - egész típusú, egyszerespontos valós és duplapontos valós - tartozó összeadó, kivonó, szorzó, osztó, összehasonlító és konvertáló rutinok felhasználását ismertetjük.

2.4.1. Egész típusú műveletek

Az egész típusú műveleteket végrehajtó rutinok operandusait a (HL), (DE) vagy a Lebegőpontos Akkumulátor #1 regiszterben kell átadni. A művelet eredménye a (HL), az A vagy a Lebegőpontos Akkumulátor #1 regiszterben keletkezik. Az alábbi táblázatban az egyes rutinok operandusait illetve a szolgáltatott eredményeket tároló regisztereket foglaljuk össze.

HL, DE regiszterben

----- A PRIMO SZÁMITÓGÉP MÓKODÉSE -----

OPERANDUS #1	OPERANDUS #2	MŰVELET	EREDMÉNY
(DE)	(HL)	+	(HL)
(DE)	(HL)	-	(HL)
(DE)	(HL)	*	(HL)
(DE)	(HL)	/	Leb. Akk. #1
(DE)	(HL)	CMP	(A)
Leb. Akk. #1	-	CINT	Leb. Akk. #1

(Az eredmény minden esetben a Leb. Akk. #1-ben is megtalálható.)

OSSZEADÁS

Belépési cím: 03026;0BD2

Egész típusú összeadás. Amennyiben az összeadás során túlcsordulás keletkezik, a rutin az összeadást mindkét operandus egyszerespontos valós típusúvá konvertálása után végzi el. Ekkor az eredmény a Lebegőpontos Akkumulátor #1-ben keletkezik.

KIVONÁS

Belépési cím: 03015;0BC7

Egész típusú kivonás. Amennyiben a kivonás során alácsordulás keletkezik, a rutin a kivonást mindkét operandus egyszerespontos valós típusúvá konvertálása után végzi el. Ekkor az eredmény a Lebegőpontos Akkumulátor #1-ben keletkezik.

SZORZÁS

Belépési cím: 03058;0BF2

Egész típusú szorzás. Amennyiben a szorzás során túlcsordulás keletkezik, a rutin a szorzást mindkét operandus egyszerespontos valós típusúvá konvertálása után végzi el. Ekkor az eredmény a Lebegőpontos Akkumulátor #1-ben keletkezik.

OSZTÁS

Belépési cím: 09360;2490

Egész típusú osztás. A műveletet csak mindkét operandus egyszerespontos valós típusúvá konvertálása után végzi el a rutin. Az eredmény a Lebegőpontos Akkumulátor #1-ben keletkezik.

OSSZEHASONLÍTÁS (CMP)

Belépési cím: 02617;0A39

Egész típusú algebrai összehasonlítás. A művelet elvégzése után a HL és DE regiszterpárok tartalma változatlan marad. Az A regiszterben képződő eredmény értelmezését a következő táblázat szemlélteti.

A tartalma	az összehasonlítás eredménye
(A)=-1	(DE)>(HL)
(A)=+1	(DE)<(HL)
(A)= 0	(DE)=(HL)

KONVERZIO

Belépési cím: 026B7;0A7F

Egyszeresponos valós, vagy duplapontos valós számot egész típusúra konvertál. A valós számot a Lebegőpontos Akkumulátor #1-ben, a szám típusát a 16559;40AF címen kell elhelyezni. Az egész típusúra konvertált szám a Lebegőpontos Akkumulátor #1-ben keletkezik. A konvertálás eredményeként a valós szám egész részének egész típusú alakját kapjuk.

2.4.2 Egyszeresponos valós műveletek

Az egyszeresponos valós műveleteket végrehajtó rutinok operandusait a BCDE és a Lebegőpontos Akkumulátor #1 regiszterben kell átadni. A műveletek eredménye az A vagy a Lebegőpontos Akkumulátor #1 regiszterben keletkezik. Az alábbiakban táblázatosan összefoglalva megadjuk az egyes rutinok operandusait, illetve az eredményt tároló regisztereket.

OPERANDUS #1	OPERANDUS #2	MŰVELET	EREDMÉNY
(BCDE)	Leb. Akk. #1	+	Leb. Akk. #1
(BCDE)	Leb. Akk. #1	-	Leb. Akk. #1
(BCDE)	Leb. Akk. #1	*	Leb. Akk. #1
(BCDE)	Leb. Akk. #1	/	Leb. Akk. #1
(BCDE)	Leb. Akk. #1	CMP	(A)
Leb. Akk. #1	-----	CSNG	Leb. Akk. #1

B = EXP
C = MSB
D = MMSB
E = LSB

OSSZEADÁS

Belépési cím: 01814;0716

Egyszeresponos valós összeadás.

KIVONÁS

Belépési cím: 01811;0713

Egyszeresponos valós kivonás.

SZORZÁS

Belépési cím: 02119;0847

Egyszeresponos valós szorzás.

----- A PRIMO SZÁMITÓGÉP MÓKODÉSE -----

OSZTÁS

Belépési cím: 02210;0BA2

Egyszeresponos valós osztás. Az osztandó értéket a BCDE regiszterekben, az osztó értéket a Lebegőponos Akkumulátor #1-ben kell elhelyezni. Az eredmény a Lebegőponos Akkumulátor #1-ben képződik.

ÖSSZEHASZNOLITÁS (CMP)

Belépési cím: 02572;0A0C

Két egyszeresponos valós szám algebrai összehasonlítása. A művelet elvégzése után a BCDE regiszterek, illetve a Lebegőponos Akkumulátor #1 értéke változatlan marad. Az összehasonlítás eredményét az A regiszter tartalmazza. Az eredmény értelmezését a következő táblázat adja meg.

A tartalma	Az összehasonlítás eredménye
(A)=-1	Leb. Akk. #1<(BCDE)
(A)=+1	Leb. Akk. #1>(BCDE)
(A)= 0	Leb. Akk. #1=(BCDE)

KONVERZIÓ 16553-et beállítani kívácsoló! Belépési cím: 02737;0AB1
 16553 nem változik

Egész típusú, ill. duplapontos valós számot egyszeresponos valós számmá konvertál. A művelet operandusa és eredménye egyaránt a Lebegőponos Akkumulátor #1-ben helyezkedik el.

2.4.3. Duplapontos valós műveletek

A duplapontos valós műveleteket végrehajtó rutinok operandusait a Lebegőponos Akkumulátor #1-ben és a Lebegőponos Akkumulátor #2-ben kell elhelyezni. A művelet eredménye vagy az A regiszterben vagy a Lebegőponos Akkumulátor #1-ben keletkezik. A következő táblázat az egyes rutinok operandusait, illetve az eredményt tároló regisztereket ismerteti.

OPERANDUS #1	OPERANDUS #2	MŰVELET	EREDMÉNY
Leb. Akk. #1	Leb. Akk. #2	+	Leb. Akk. #1
Leb. Akk. #1	Leb. Akk. #2	-	Leb. Akk. #1
Leb. Akk. #1	Leb. Akk. #2	*	Leb. Akk. #1
Leb. Akk. #1	Leb. Akk. #2	/	Leb. Akk. #1
Leb. Akk. #1	Leb. Akk. #2	CMP	(A)
Leb. Akk. #1	----	CDBL	Leb. Akk. #1

----- A PRIMO SZÁMITÓGÉP MÓKODÉSE -----

OSSZEADÁS

Belépési cím: 03191;0C77

Duplapontos valós összeadás.

KIVONÁS

Belépési cím: 03184;0C70

Duplapontos valós kivonás.

SZORZÁS

Belépési cím: 03489;0DA1

Duplapontos valós szorzás.

OSZTÁS

Belépési cím: 03557;0DE5

Duplapontos valós osztás. Az osztandót a Lebegőpontos Akkumulátor #1-be, az osztót a Lebegőpontos Akkumulátor #2-be kell elhelyezni. A hányados a Lebegőpontos Akkumulátor #1-ben képződik.

OSSZEHASONLÍTÁS (CMP)

Belépési cím: 02680;0A78

Két duplapontos valós szám algebrai összehasonlítása. A művelet elvégzése után mindkét Lebegőpontos Akkumulátor értéke változatlan marad. Az összehasonlítás eredményét az A regiszter tartalmazza, melynek értelmezését a következő táblázat adja meg.

A tartalma	Az összehasonlítás eredménye
(A)=-1	Leb. Akk. #1 > Leb. Akk. #2
(A)=+1	Leb. Akk. #1 < Leb. Akk. #2
(A)= 0	Leb. Akk. #1 = Leb. Akk. #2

KONVERZIÓ

Belépési cím: 02779;0ADB

Egész típusú, ill. egyszeres pontos valós számot duplapontos valós számmá konvertál. A konverzió operandusa és eredménye egyaránt a Lebegőpontos Akkumulátor #1-ben helyezkedik el.

2.5. Függvények

Az alábbiakban ismertetjük az aritmetikai függvények felhasználásához szükséges paramétereket és a függvényszámítás algoritmusát. Megadjuk az egyes függvényeket megvalósító rutinok decimális és hexadecimális belépési cimeit. A függvényértékek egy részét a PRIMO a matematikából ismert sorfejtésre alapozva határozza meg. Az adott sor konvergenciájától függően a duplapontos számoknál szokásos pontosság eléréséhez a sor sokkal több tagját

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

kellene kiszámítani, ami az aritmetika működési idejét tekintve gazdaságtalan lenne. Ezért a PRIMO - más gépekhez hasonlóan - csak egyszerespontos valós értéket szolgáltat a logaritmikus, exponenciális, trigonometrikus és inverz trigonometrikus függvények esetén. A függvények argumentumát a Lebegőpontos Akkumulátor #1-ben kell elhelyezni. Ha a függvény az előbb említett, csak egyszerespontos valós értéket adó függvények valamelyike, akkor az argumentumnak egyszerespontos valós számnak kell lennie. A függvényértéket a Lebegőpontos Akkumulátor #1-ben kapjuk vissza.

SIN(x)

Belépési cím: 05447;1547

A függvényértéket a PRIMO a

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!}$$

sor alapján határozza meg, ahol x radiánban értelmezett szög.

COS(x)

Belépési cím: 05441;1541

A függvényértéket a PRIMO az ismert

$$\cos(x) = \sin(x + \pi/2)$$

azonosság alapján határozza meg. A kifejezésben x radiánban értelmezett szög.

TAN(x)

Belépési cím: 05544;15A8

A függvényérték meghatározására a PRIMO a

$$\tan(x) = \sin(x) / \cos(x)$$

összefüggést használja fel. A kifejezésben x radiánban értelmezett szög.

ATN(x)

Belépési cím: 05565;15BD

A megfelelő pontosság eléréséhez a PRIMO az

$$\operatorname{atn}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \frac{x^{11}}{11} + \frac{x^{13}}{13} - \frac{x^{15}}{15} + \frac{x^{17}}{17}$$

sor aktuális értékét határozza meg.

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

EXP(x)

Belépési cím: 05177;1439

A függvényértéket a PRIMO következő sor értékének meghatározásával számítja ki.

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!}$$

LOG(x)

Belépési cím: 02057;0809

A függvényérték kiszámításához x értékének alkalmas átalakítása és a

$$0.5 \leq x < 1$$

intervallumba történő normalizálása után a PRIMO a következő sort használja fel.

$$\log(x) = 2 * \left(\frac{x-1}{x+1} \right) + \frac{2}{3} * \left(\frac{x-1}{x+1} \right)^3 + \frac{2}{5} * \left(\frac{x-1}{x+1} \right)^5$$

x^y

Belépési cím: 05106;13F2

A hatványérték meghatározásakor a PRIMO az

$$x^y = e^{y * \ln(x)}$$

azonosságot használja fel. A rutin meghívása előtt x egyszerűes pontos valós értékét a STACK-be kell elhelyezni úgy, hogy a STACK tetején a szám Kitevője, alatta a szám MSB Byte-ja, stb. álljon. A tetszőleges típusú y értéknek a Lebegőpontos Akkumulátor #1-ben kell lennie.

SQR(x)

Belépési cím: 05095;13E7

A függvény értékének meghatározására a PRIMO a következő azonosságot alkalmazza.

$$\text{sqr}(x) = e^{0.5 * \ln(x)}$$

RND(x)

Belépési cím: 05321;14C9

A PRIMO egy véletlen érték meghatározására az előzőleg előállított véletlen számot használja fel. Az alkalmazott képlet a következő:

$$x_{i+1} = (x_i * 5105216) \bmod 2^{24} + 372837$$

be: #1 tip: 2,4,8

Ki: #1 tip 4 v. 2?

----- A PRIMO SZAMITOGEP MOKODESE -----

Ha az RND függvény hívásakor a Lebegőpontos Akkumulátor #1 értéke 0...1 intervallumba esik, akkor a rutin által előállított véletlen szám is ezen intervallumon belül található. Amennyiben a híváskor átadott paraméter egynél nagyobb szám volt, a generált véletlen érték a 0...paraméter érték közé eső egész szám lesz.

INT(x)

Belépési cím: 02871;0B37

Az INT függvény eredményül a Lebegőpontos Akkumulátor #1-ben őrzött számértékhez legközelebb eső, annál kisebb, vagy azzal egyenlő egész számot adja.

FIX(x)

Belépési cím: 02854;0B26

A függvényérték kiszámítása a következő képlet alapján történik.

$$\text{fix}(x) = \text{sgn}(x) * \text{int}(\text{abs}(x))$$

ABS(x)

Belépési cím: 02423;0977

A függvény a Lebegőpontos Akkumulátor #1-ben őrzött szám abszolút értékét adja eredményül.

SGN(x)

Belépési cím: 02442;098A

Az SGN függvény eredményül egy egész típusú számot szolgáltat. Az eredmény értéke a Lebegőpontos Akkumulátor #1-ben őrzött szám előjelének függvényében -1, 0, ill. +1 lesz.

3. FEJEZET

A PRIMO periféria kezelő rutinjai

Ebben a fejezetben ismertetjük a PRIMO periféria kezelő programjait. Megadjuk az egyes rutinok ugrótáblában elhelyezett kezdőcímét, felsoroljuk a hívás előtt beállítandó paramétereiket. Annak érdekében, hogy a rutinok részletes működése iránt érdeklődők igényeit is kielégítsük, e fejezetben tárgyalt valamennyi rutin teljes - megjegyzésekkel kiegészített - programlistája a Függelékben megtalálható.

3.1. A periféria kezelő rutinok ugrótáblája

Egy számítógép - köztük a PRIMO - fejlesztése, ill. továbbfejlesztése hosszabb időt vesz igénybe. A fejlesztéseknek azonban mindig figyelembe kell venniük a korábbi állapotokat is. Annak érdekében, hogy a berendezés korábbi változatához elkészített programok az újabb verzió megjelenésével - lehetőleg változtatás nélkül, vagy minimális átalakítással - felhasználhatók maradjanak, igen gyakran a működtető programokban u.n. ugrótáblát hoznak létre. Ennek célja, hogy a leggyakrabban alkalmazott szubrutinok memórián belüli címe a számítógép fejlesztése során szabadon megváltozhasson anélkül, hogy ez a már elkészített programok módosítását tenné szükségessé. Az ugrótábla nem más, mint a memória rögzített címén elhelyezett ugró - feltétlen vezérlésátadó - utasítások sorozata. Az ugrótábla minden utasítása egy-egy gyakran használt szubrutinra adja át a vezérlést. Az ugró utasítások sorrendjét a számítógép első változatában rögzítik, így az a rendszert működtető programok változtatásakor nem módosul, csak az ugrási címek változnak az új verzióknak megfelelően. Ha a Felhasználó az egyes rutinokat az ugrótáblán keresztül hívja meg, a rendszerprogram változása miatt szükségessé vált címmódosítások nem befolyásolják programja működését.

Mivel a változtatás lehetőségét a PRIMO esetében is biztosítani akartuk, a PRIMO is rendelkezik egy ugrótáblával. Ebbe a táblába a periféria kezelő programokra ugró utasításokat gyűjtöttük össze. A fejezet hátralévő részében ismertetésre kerülő rutinok - néhány kivételtől eltekintve - az ugrótáblán keresztül elérhetők. Ha a Felhasználó nem sérti meg az ugrótábla elvet, programjai a PRIMO továbbfejlesztett változatain is működőképesek maradnak.

3.2. NMI/TIMER rutin

A PRIMO működését minden 1/50-ed másodpercben egy NMI - Nem Maszkolható Interrupt - szakítja meg. Valahányszor a Z-80 mikroprocesszor az NMI lábon aktív szintet érzékel, felfüggeszti az éppen futó program végrehajtását, és a vezérlést a 102;66 címre adja. Itt található az NMI/TIMER rutin első utasítása.

Az NMI/TIMER rutin kettős feladatát lát el. Egyrészt a memória 16445;403D...16447;403F címeken elhelyezkedő 24 bites számláló értékét növeli, másrészt a PRIMO hátoldalán lévő RESET nyomógomb állapotát vizsgálja.

- A 24 bites számláló lehetővé teszi, hogy a PRIMO 1/50 másodperc pontossággal ~24 órányi időtartamot mérjen. Mivel a számláló értékét az NMI/TIMER rutin növeli, függetlenül attól, hogy a PRIMO BASIC, vagy gépi kódú programot hajt-e végre, az 1/50 másodpercenként esedékes változtatás bekövetkezik. A számláló módosítása még a magnóra történő írás/olvasás idején sem szünetel.
- A számláló értékének módosítása után a PRIMO megvizsgálja a RESET nyomógomb állapotát. Ha úgy találja, hogy a gombot nem nyomják, az NMI/TIMER rutin befejezi működését és a vezérlést a RAM 16408;4018 címén elhelyezkedő utasításra adja. Itt rendszeren egy olyan ugró utasítás található, amely a vezérlést visszaadja a megszakított programnak. Ha azonban a Felhasználó ezt megváltoztatja, akkor az NMI/TIMER rutint saját gépi kódú programjával egészítheti ki. Ha a PRIMO a RESET gombot megnyomott állapotban találja, megvárja míg azt elengedik, majd a vezérlést a RESET rutinra adja.

Az NMI/TIMER rutin csak akkor működik, ha az NMI engedélyezett. Azt ugyanis a KI-1 kimeneti kapu megfelelő programozásával ki lehet kapcsolni.

3.3. RESET rutin

A RESET rutin feladata, hogy a PRIMO periféria kezelő rutinjainak RAM-ban elhelyezkedő munkaregisztereit alaphelyzetbe állítsa. Ez két esetben történik meg:

- a PRIMO bekapcsolásakor, ill. abban az esetben, ha a vezérlés a 0 címre adódik. A munkaregiszterek inicializálása után a PRIMO megvizsgálja, hogy a ROM 2.-4. tokja a '84.1 verzió programjait tartalmazza-e. Ha "idegen" tokot talál, ahelyett, hogy a BASIC Interpretert indítaná, a vezérlést az "idegen" toknak adja. (A ROM-csere megvalósításának részleteivel a PRIMO Hardver Leírás foglalkozik.)

- a RESET billentő megnyomása, majd elengedése után, ha az NMI engedélyezve van. Ilyenkor a munkaregiszterek inicializálása után a PRIMO a vezérlést a RAM 16414;401E címén elhelyezkedő utasításra adja. Itt rendszeren egy olyan ugró utasítás található, amely a vezérlést a BASIC Interpreter direkt - parancs - üzemmódját megvalósító rutin kezdetére adja. Ha azonban a Felhasználó ezt megváltoztatja, akkor a RESET rutint saját gépi kódú programjával egészítheti ki.

3.4. Várakozás képkiojtásra rutin - VBLANK

Ha a képernyőn - akár egy rajzfilmen - egy alakzatot kívánunk mozgatni, gondoskodnunk kell arról, hogy a képtartalom csak a képkiojtás ideje alatt változzék. Ha ezt nem tesszük meg, a képernyőn - szándékunk ellenére - villódzó ábrát láthatunk.

A VBLANK rutin meghívása után kivárja, amíg a PRIMO kiojtja a hozzá kapcsolt TV képernyőjét, majd visszatér a hívó programba. Ekkor a Felhasználó a szükséges képtartalom módosításokat a képkiojtás hátralévő idejében - ~7 msec - elvégezheti.

Belépési cím: ~~00101,0001~~ 12449;

(A rutin kezdőcíme - mivel nem ugrotábla-cím a PRIMO '84.1 verziójára érvényes.)

3.5. Képernyő kezelő rutin - DSPHND

A képernyő kezelő rutin feladata, hogy a PRIMO képernyőjére normál, ill. nyújtott, vízszintes vagy függőleges állású karaktereket rajzoljon. Ezen túlmenően a DSPHND végrehajtja a vezérlő kódok - 1...29 - által kijelölt műveleteket is.

A képernyő kezelő program számára a megjelenítendő/vezérlő kódot az A regiszterben kell átadni. A DSPHND működésének kezdetén menti a mikroprocesszor regisztereinek állapotát, majd eldönti, hogy a kód megjeleníthető karaktert, ill. vezérlő funkciót jelez-e. A vizsgálat eredményének függvényében a rutin különbözőképpen folytatja működését.

- Ha a kód egy vezérlő karakter, a DSPHND meghatározza, hogy a lehetséges 24 funkció közül éppen melyiket kell végrehajtania. Ezután a vezérlést a szubrutin azon része kapja, amely a kijelölt művelet végrehajtására íródott. A részprogram működésének végezetével megtörténik a korábban mentett regiszter tartalmak visszaállítása, majd a vezérlést a hívó program kapja vissza.
- Amennyiben bemeneti paraméterként egy megjeleníthető karakter kódját adtuk meg, a DSPHND megállapítja, hogy Felhasználó

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

ló által definiált - 128...255 - vagy normál - 30...127 - karaktert kell-e a képernyőre rajzolni. A vizsgálat eredményének függvényében a PRIMO a karakter pontmintázatát vagy a Kiegészítő Karaktergenerátor Területről - melynek kezdőcímét a RAM 16459;404B címén kezdődő szó tárolja - vagy az Alap Karaktergenerátor Területről olvassa ki. Miután a DSPHND meghatározta a pontmintázat címét, megkezdődik a karakter megjelenítése. Ennek során a RAM 16455;4047 címén elhelyezkedő állapotjelző byte pillanatnyi tartalma által meghatározott formában a DSPHND a kurzor aktuális pozíciójába rajzolja a kiválasztott karaktert. A megjelenítés után beállítja a kurzor új pozícióját. Ekkor - ha szükséges - elvégzi a képernyő tartalom egy sorral feljebb csúsztatását - roll - is. A karakter megjelenítése után megtörténik a korábban mentett regiszterek tartalmának visszaállítása, majd a vezérlést a hívó program kapja vissza.

(A megjelenítendő karaktert a DSPHND a kurzor aktuális pozíciójába rajzolja. E karakterpozíció bal felső pontjának X és Y irányú koordinátáit a RAM 16456;404B és 16457;4049 címein elhelyezkedő byte-ok tárolják. Mivel ezek - a lehetséges tartományon belül - tetszőleges értéket felvehetnek, a megjelenő karakter bal felső pontja a képernyő 49152 pontja közül bármelyik lehet. Így az is előfordulhat, hogy pl. vízszintes írás esetén a karakter alsó, ill. jobb oldali pontjai már nem férnek el a képernyőn. Ezt a PRIMO felismeri, és nem folytatja a karakter rajzolását. [Ellenkező esetben a túlnyúló pontok a kép bal-, ill. felső szélén jelennének meg.] Ez viszont azt eredményezi, hogy a képernyőn csonka karakter jelenik meg. Az említett eset csak akkor fordulhat elő, ha a Felhasználó a kurzor pozíció X és Y koordinátáit meggondolatlanul - pl. a BASIC POKE utasításának felhasználásával - változtatta meg, ill. a vezérlő kódokat figyelmetlenül alkalmazta. Mindkét esetben a képernyő törlése - 12;0C kód - visszaállítja a helyes X és Y koordinátákat.)

Belépési cím: 00021;0015

Bemeneti paraméterek:

(A) = megjelenítendő/végrehajtandó kód

Kimeneti paraméterek:

A DSPHND rutin a mikroprocesszor valamennyi regiszterének eredeti állapotát megőrzi. Visszatérés után a flag-ek állapota az A regiszter tartalmának megfelelő.

3.6. Billentyűzet kezelő rutinok

A PRIMO billentyűzet kezelő rutinjainak célja, hogy egy megérintett gomb ASCII kódját előállítsák, és a hívó program rendelkezésére bocsássák.

A két rutin a billentyűzet vizsgáló - KYBHND - és a kivárássos billentyűzet vizsgáló - KKBHND -. A következőkben külön-külön összefoglaljuk az egyes rutinok tevékenységeit.

- A KYBHND meghívása után menti a mikroprocesszor regisztereit, majd megvizsgálja, hogy nyomják-e a billentyűzet valamelyik gombját. Ha aktiv billentyűt talál, egy táblázat segítségével meghatározza annak ASCII kódját, majd a korábban mentett regiszter tartalmak visszaállítása és az előállított kód A regiszterbe helyezése után visszatér a hívó programba. Amennyiben a billentyűzet egyetlen gombját se találja aktívnak, az A regiszterben 0 értékkel tér vissza a hívó programba. (A KYBHND rutin u.n. speciális üzemmódban is üzemeltethető. Ezt a RAM 16453;4045 címén elhelyezkedő byte 0-as bitjébe írt 1 értékkel válthatjuk ki. Ekkor a ↓ billentyű speciális jelentést kap. Egy gomb és a ↓ billentyű együttes megérintése - a SHIFT gombhoz hasonlóan - módosítja a rutin által visszaadott ASCII kódot. Ennek eredményeként nem a gomb eredeti értéke, hanem annál 128-al nagyobb kód kerül az A regiszterbe.)

Belépési cím: 00029;001D

Kimeneti paraméterek:

A KYBHND rutin a mikroprocesszor valamennyi regiszterének eredeti állapotát megőrzi. Visszatérés után a flagok állapota a visszaadott kód értékének - az A regiszter tartalmának - megfelelő.

- A KKBHND meghívása után menti a mikroprocesszor regisztereit, majd meghívja a KYBHND rutint. Ha úgy találja, hogy a visszakapott kód nullától különbözik - nyomják valamelyik billentyűt -, a korábban mentett regiszter tartalmak visszaállítása és a KYBHND-től kapott kód A regiszterbe helyezése után visszatér a hívó programba. Amennyiben a visszakapott kód tanúsága szerint a billentyűzet egyetlen gombját se nyomják, mindaddig újra indítja a KYBHND rutint, amíg az aktiv billentyűt nem talál. Ennek megfelelően a KKBHND rutin csak akkor tér vissza a hívó programba, ha megérintjük a PRIMO valamelyik gombját. A várakozás ideje alatt a képernyőn az aktuális pozícióban egy villogó kurzor látható.

----- A PRIMO SZÁMÍTÓGÉP MŰKÖDÉSE -----

Amennyiben egy gombot megérintünk, a KKBHND a megszerzett kóddal visszatér a hívó programba. Ha az ismét elindítja a kivárásos billentyűzet kezelő rutint és az előbbi gombot megérintése óta folyamatosan nyomjuk, egy megfelelő időzítés letelte után - annak ellenére, hogy a gomb állapotában változás nem történt - a vezérlést ismét a hívó program kapja vissza. Ezzel a KKBHND rutin megvalósítja az u.n. automatikus ismétlés - auto repeat - funkciót is.

Belépési cím: 00037;0025

Kimeneti paraméterek:

A KKBHND rutin a mikroprocesszor valamennyi regiszterének eredeti állapotát megőrzi. Visszatérés után a flag-ek állapota a visszaadott kód értékének - az A regiszter tartalmának - megfelelő.

3.7. Nyomtató kezelő rutin - PRTHND

A PRTHND rutin feladata, hogy a PRIMO-hoz illesztett nyomtatóra egy karaktert írjon ki. A nyomtató kezelő rutin végrehajtja - a beállítható paramétereknek megfelelően - a sorok és lapok tördelését is.

~~A PRTHND rutin meghívása után menti a mikroprocesszor regisztereit, majd megvizsgálja, hogy az A regiszterben kapott karakter vezérlő kód-e. A vizsgálat eredményének függvényében a rutin működése a következőképpen alakul.~~

- Ha a vizsgált karakter nem vezérlő kód, a PRTHND a karaktert a nyomtatóra listázza. Ennek során először megvizsgálja, hogy a nyomtató képes-e egy byte fogadására. Ha azt találja, hogy a rendszerben nincs üzemképes nyomtató - nincs illesztve, vagy nincs bekapcsolva -, hangjelzéssel figyelmezteti a Felhasználót. Ha van üzemképes nyomtató, de az éppen egy korábban kiküldött karakter papírra írásával van elfoglalva, kivárja amíg az befejeződik, majd elküldi a karakter kódját a nyomtatónak. Ezután a PRTHND megvizsgálja, hogy az éppen nyomtatás alatt álló sor hossza elérte-e a RAM 16464;4050 címén tárolt maximális sorhossz értékét. Ha nem, a korábban mentett regiszter tartalmak visszaállítása után a vezérlést a hívó program kapja meg. Amennyiben a sor elérte az engedélyezett hosszt, a PRTHND egy CR - Carriage Return, Kocsi Vissza - karaktert küld a nyomtatóra, majd megvizsgálja, hogy a lapon lévő sorok száma elérte-e a RAM 16465;4051 címén tárolt maximális laphossz értékét. Ha nem, a korábban mentett regiszter tartalmak visszaállítása után a vezérlést a hívó program kapja vissza. Amennyiben betelt a lap, a PRTHND egy FF - Form Feed, Lapdobás - karaktert küld a

----- A PRIMO SZÁMÍTÓGÉP MŰKÖDÉSE -----

nyomtatóra. (Abban az esetben, ha a 16465;4051 cím értéke 127-nél nagyobb, a PRTHND nem vizsgálja a lap beteltét, vagyis nem küld ki FF kódot.) A PRTHND kezdetben egy sor maximális hosszát 80-nak, egy lapon lévő sorok maximális számát 58-nak tekinti.

- A PRTHND két vezérlő kódot képes értelmezni. Az egyik a CR, a másik a FF. Ha e kódok valamelyikét felismeri, úgy jár el, amint azt a karakter kinyomtatását követő tevékenységeknél az előbbieken leírtuk.

Az eddig említetteken túl a PRTHND arra is utasítható, hogy a PRIMO két karakterének kódját - a különféle nyomtatók egyszerűbb illesztése érdekében - átalakítsa. Ezek a 30;1E - i - és 31;1F - † - kódok. Ha a RAM 16453;4045 címén elhelyezkedő byte 3-as bitjébe 0 értéket írunk, a PRTHND az említett kódok előfordulása esetén a következő konverziókat hajtja végre:

30;1E - i - → 105;69 - i -
31;1F - † - → 94;5E - ASCII † -

Alapértelmezésben a PRTHND végrehajtja a konverziót.

Belépési cím: 00045;002D

Bemeneti paraméterek:

(A) = a kinyomtatandó/végrehajtandó kód

Kimeneti paraméterek:

A PRTHND rutin a mikroprocesszor valamennyi regiszterének eredeti állapotát megőrzi. Visszatérés után a flag-ek állapota a kiíratott karakter kód értékének - az A regiszter tartalmának - megfelelő.

3.8. Egy sort beolvasó rutin - GLINE

A BASIC Interpreter működése során - de bármely más, a Felhasználó által írt programban is - gyakran szükséges, hogy a PRIMO a billentyűzetről több karakterből álló stringet olvasson be. A GLINE rutin célja, hogy végrehajtsa ezt az ismétlődő tevékenységet.

A GLINE által beolvasható sor maximálisan 210 karaktert tartalmazhat. A rutin a beolvasott kódokat beérkezésük sorrendjében eltárolja egy - szabadon megválasztható címen elhelyezkedő - pufferbe. A sor beolvasó rutin a karakter kódok tárolásával egyidőben azokat a képernyőn is megjeleníti - echo -. A rutin meghívása után menti a mikroprocesszor megfelelő regisztereit,

----- A PRIMO SZÁMÍTÓGÉP MŰKÖDÉSE -----

majd meghívja a KKBHND szubrutint. A kivárásos billentyűzet kezelő rutintól visszakapott kódról eldönti, hogy vezérlő-, ill. megjeleníthető karakter kódja-e. Annak függvényében, hogy mi a vizsgálat eredménye, a GLINE rutin különféleképpen működik.

- Ha a kód egy megjeleníthető karaktert jelöl, a GLINE rutin megvizsgálja, hogy az eddig pufferbe tárolt karakterek száma egyenlő-e 210-zel. Ha már teljesen feltöltötte a tárolót, az új kód pufferbe írása helyett hangjelzéssel figyelmezteti a Felhasználót a sor beteltére. Amennyiben még nincs 210 byte a pufferben, elhelyezi az újonnan beolvasott karakter kódot a tároló soronkövetkező szabad címére, majd az új állapotnak megfelelően módosítja a puffer pointer és a sorhossz számláló értékét. Ha ezeket a feladatokat végrehajtotta, meghívja a DSPHND rutint, amellyel a beolvasott kódnak megfelelő karaktert a képernyőre rajzoltatja. Ezután a GLINE rutin működése egy újabb karakter beolvasásával folytatódik.
- Ha a beolvasott kód egy vezérlő karakter, akkor megvizsgálja, hogy az CR, BRK, SHIFT/→, ←, SHIFT/←, vagy CLS kódja-e. Amennyiben az érték valamennyi vizsgált kódtól különböző, anélkül, hogy a vezérlő kódot a pufferbe írná, a GLINE rutin meghívja a képernyő kezelő szubrutint, amely végrehajtja a megadott vezérlő kód által kijelölt funkciót. Ezután a GLINE rutin működése egy újabb karakter beolvasásával folytatódik.

Ha a vizsgált érték a ← billentyű kódja - OB, BACK STEP -, a rutin kitörli a pufferből az oda utoljára rögzített kódot, majd a puffer pointer és a sorhossz számláló értékét az új helyzetnek megfelelően módosítja. Ezzel egy időben törli a képernyőről az utoljára megnyomott billentyűhöz rendelt karakter képét.

Ha a GLINE SHIFT/←, vagy CLS kódot talál, törli az eddig rögzített összes karakter kódját a pufferből, és eltávolítja a karakterek képét a képernyőről - sor törlés -. Ezután előről kezdhető a karaktersorozat begépelése.

Ha a beérkezett kód egy vízszintes tabulátor - SHIFT/→ - kódja, a rutin a karaktert figyelmen kívül hagyja, és folytatja a karakterek beolvasását.

A GLINE rutin befejezi a puffer feltöltését, ha CR, vagy BRK kódot észlel. Ilyenkor a puffer végére egy 0 értékű byte-ot helyez, majd a képernyőn a kurzort - a DSPHND rutin segítségével - a következő sor elejére állítja. Ezután a vezérlést visszaadja a hívó programba. Ott a CARRY bit vizsgálatával eldönthető, hogy a sor utolsó karaktere CR, vagy BRK volt-e.

CY=0 CY=1

A GLINE rutin a KKBHND szubrutint speciális módban futtatja.

Ez azt eredményezi, hogy lehetővé válik - a 3.6. fejezetben leírt módon - 128...255 közé eső kódok begépelése is. Ha a **GLINE** ilyen tartományba eső kódot észlel, mielőtt azt a pufferbe helyezné, a vezérlést a RAM 16423;4027 címére adja. Itt rendszeren egy JP utasítás található, amely visszaadja a vezérlést a **GLINE** rutinba. Ha ezt a Felhasználó átírja, lehetővé válik, hogy a speciális kód feldolgozásáról maga gondoskodjék. (Lehetséges pl. olyan program készítése, amely egyetlen billentyű megnyomására egész szó - pl. egy **BASIC** kulcsszó - bevitelét eredményezi.)

Belépési cím: 00053;0035

Bemeneti paraméterek:

(HL) = sort tároló puffer kezdőcíme - 1

Kimeneti paraméterek:

(HL) = sort tároló puffer kezdőcíme - 1

(CY) = 1, ha a sor végén **BRK** volt

3.9. Magnó kezelő rutinok

A következőkben ismertetjük a magnó kezelést végrehajtó rutinok működését. Kitérünk az állományok logikai felépítésére, valamint a magnó fizikai kezelésére. Megadjuk az állománykezeléshez szükséges rutinok belépési címeit és paramétereit.

3.9.1. Logikai állománykezelés

Akár adatokat, akár programokat, vagy a képernyő aktuális tartalmát rögzítjük a magnóra, a kívitt információ minden esetben egy logikailag összetartozó adathalmazt - u.n. állományt, idegen szóval file-t - alkot. Az állomány egy névvel ellátott, meghatározott szerkezetű jelsorozat. Az állomány kisebb logikai egységekre - u.n. blokkokra - bontható. A blokkok némelyike speciális - az állományra vonatkozó - információt tartalmaz, a többi adatokat tartalmazó adat-blokk. Blokk alatt általában meghatározott hosszúságú, ismert felépítésű adatsorozatot értünk.

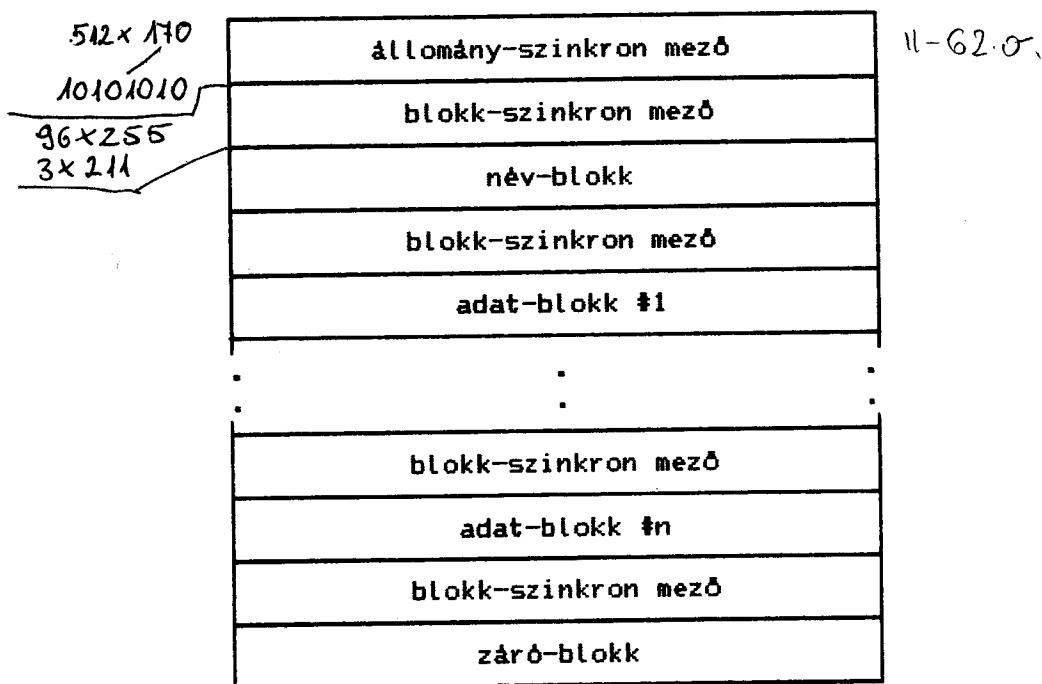
Az alábbiakban ismertetjük az állományok, ill. a blokkok típusait és felépítésüket.

Minden állomány három különböző típusú blokkból épül fel. Az állomány első blokkjának minden esetben az állomány nevét tartalmazó név-blokknak kell lennie. Ezt tetszőlegesen sok adat-blokk követheti. Az állomány végén u.n. záró-blokk áll. Annak érdekében, hogy a magnóról az állományokat a **PRIMO** biztonságosan vissza tudja olvasni, minden állomány név-blokkja előtt egy u.n. álló-

----- A PRIMO SZÁMITÓGÉP MÓKODÉSE -----

mány-szinkron mező helyezkedik el. A beolvasás megbízhatóságának további fokozása érdekében minden blokk előtt egy u.n. blokk-szinkron mező található.

Az eddig leírtak alapján egy állomány általános felépítése:



Az állomány tehát különféle típusú blokkok sorozata. A blokkok szabványos felépítésűek, közös jellemzőik a következők:

- Minden blokk elején a már említett blokk-szinkron mező található. Ez 96 db 255;FF értékű byte, amelyet 3 db 211;D3 értékű szinkron byte zár le. A PRIMO akkor tudja a blokk szinkronizációt helyesen végrehajtani, ha a három 211;D3 értékű szinkron byte közül legalább az egyiket felismeri.
- A blokk első - szinkron byte-okat követő - karaktere az 1 byte hosszúságú blokk-típus. Ez az éppen betöltés alatt álló blokk típusát jelzi. (A lehetséges blokk-típusokat később ismertetjük.)
- A blokk második - blokk-típust követő - karaktere az adott blokk állományon belüli sorszámát tartalmazza. A blokk-sorszám 1 byte hosszú, benne BCD kódban egy 0...99 közé eső decimális szám található.
- A blokk-típus és a blokk-sorszám után következnek az adatokat hordozó karakterek. Ezek száma és értelmezése az adott

blokk típusától függ.

- A blokk utolsó karaktere az ellenőrző-összeg. Ez a blokk átvitele során fellépő esetleges hibák felismerését szolgálja. Az ellenőrző-összeg egy 8 bites érték, amely a blokkban lévő - a blokk-típust követő - valamennyi byte értéke összegének 256-tal történő osztása után kapott maradékkal egyenlő.

A blokkok általános felépítésének leírása után a következőkben az egyes blokk típusok feladatát és felépítését ismertetjük.

Név-blokk

Feladata: Az állomány elejének jelzése, az állomány azonosítását célzó - maximálisan 16 karakter hosszúságú - állománynév tárolása.

Felépítése:

1. byte - blokk-típus (1 byte)
 - 131;83 - BASIC program név-blokk
 - Az állomány vegyesen BASIC, vagy gépi kódú program-blokkokat, ill. Képernyő tartalom-blokkokat tartalmazhat.
 - 135;87 - Adat-állomány
 - Az állomány csak BASIC adat-blokkokból állhat.
2. byte - blokk-sorszám (1 byte)
 - értéke kötelezően 0.
3. byte - Állománynév hossz (1 byte)
 - értéke az állomány nevének hosszától függően 1...16 lehet.
4. - Állománynév (1...16 byte)
 - Tetszőleges karaktereket tartalmazó 1...16 byte hosszúságú karaktersorozat.
- n. byte - ellenőrző-összeg (1 byte)

Adat-blokk

Feladata: Tetszőleges bináris adatsorozat tárolása.

Felépítése:

1. byte - blokk-típus (1 byte)
 - 241;F1 - BASIC program-blokk
 - 245;F5 - Képernyő tartalom-blokk
- a PR. kezdő+ utolsó névvel helyre
+ 64600k*

----- A PRIMO SZÁMITÓGÉP MÓKODÉSE -----

- 247;F7 - BASIC adat-blokk
- 249;F9 - Gépi kódú program-blokk
- 2. byte - blokk-sorszám (1 byte)
értéke 0...99 közötti BCD szám.
- 3. byte - betöltési cím (2 byte)
A betöltési cím BASIC program-blokk, BASIC adat-blokk és Képernyő tartalom-blokk esetén relatív cím (a blokkban lévő 1. byte állományon belüli sorszáma), Gépi kódú program-blokk esetén abszolút cím (a blokkban lévő 1. byte betöltés utáni memória címe).
- 5. byte - blokkban lévő byte-ok száma (1 byte)
A byte-szám 0...255 közötti értéket vehet fel. Ha a számláló 0 értékű, ez 256 db byte-ból álló blokkot jelez.
- 6. ... - Adat byte-ok (1...256 byte)
- n. byte - ellenőrző-összeg (1 byte)

Záró-blokk

Feladata: Az állomány végének jelzése.

Felépítése:

- 1. byte - blokk-típus (1 byte)
 - 177;B1 - BASIC-, vagy Gépi kódú program-állomány vég. A záró-blokk felismerése után a PRIMO a vezérlést a BASIC Interpreternek adja vissza.
 - 181;B5 - Képernyő tartalom-állomány vég. A záró-blokk felismerése után a PRIMO a vezérlést a BASIC Interpreternek adja vissza.
 - 183;B7 - BASIC adat-állomány vég. A záró-blokk felismerése után a PRIMO a vezérlést a BASIC INPUT# utasítását végrehajtó rutinnak adja vissza.
 - 185;B9 - Automatikusan induló Gépi kódú program-állomány vég. A záró-blokk felismerése után a PRIMO a vezérlést a záró-blokk hátralévő részében megadott memória címre adja át.
- 2. byte - blokk-sorszám (1 byte)
- 3. byte - indítási-cím (2 byte)
Indítási-cím csak a 185;B9 típusú blokkban adható meg. Más típusú blokk esetén nincs indítási-cím mező.
- n. byte - ellenőrző-összeg (1 byte)

Az állományok blokkokra tördelésének előnye, hogy betöltési hiba - bit tévesztés - esetén nem szükséges a teljes állomány ismételt beolvasása, elegendő a hibás blokk újraolvasása. A blokk struktúra következtében ugyanis egy hibás blokk nincs hatással a következő blokkra - és így az állomány hátralévő részére -, hanem a következő blokk elején álló blokk-szinkron mező segítségével a PRIMO működését újra szinkronizálhatja. Ezzel az adatrögzítés biztonsága jelentősen fokozható.

3.9.2. A magnó fizikai kezelése

A logikai állománykezelés áttekintése után a következőkben a magnó fizikai kezelését ismertetjük. Ebben a fejezetben írjuk le a blokkokat alkotó byte-ok felírásának és visszaolvasásának menetét, a megbízható adatátvitelt segítő szinkronizációval kapcsolatos részleteket.

Mivel a magnóra az adatok jelsorozat formájában kerülnek, így a rögzítési elv ismertetésekor elegendő egyetlen bit felírásának mikéntjét bemutatni.

Egy bit felírása a magnóra

A személyi számítógépekhez általában kiskereskedelmi forgalomban kapható magnókat alkalmaznak. Ezek többnyire csak a hangfrekvenciás tartomány középső részén, kb. 400...6000 Hz között képesek a jeleket elfogadható formában rögzíteni. Még e szűk tartományon belül is igen sok problémát okoz a gyengébb készüléknél tapasztalható magas zajszint és az igen nagy torzítás. (Általánosságban elmondható, hogy megbízható adatrögzítésre csak jól beállított, gondosan karbantartott magnó használható. A berendezés olyan hibája, amelyet zenehallgatás során még csak tapasztalt megfigyelő ismer fel, már megghiusíthatja az adatok helyes tárolását. Ezért elsősorban a magnó író/olvasó fejének tisztaságára, az egyenletes szalagsebességre, valamint a megfelelő bemeneti/kimeneti feszültségszintek beállítására kell fokozottan ügyelni.)

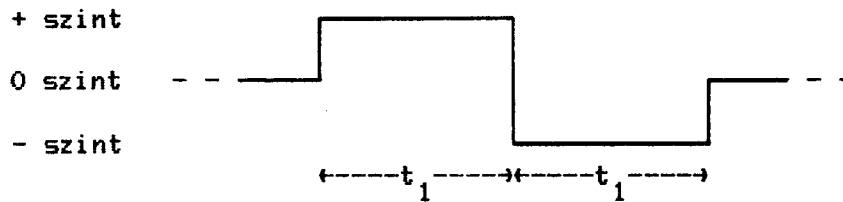
A digitális jelrögzítés alapgondolata, hogy a 0, ill. 1 értékű bitekhez két - egymástól viszonylag távoli, de a magnó biztonságos átviteli tartományába eső - különböző frekvenciájú jelet rendelünk. Visszaolvasáskor a beérkező jelek periódusidejének mérésével meghatározható, hogy a soronkövetkező bit 0, ill. 1 értéket tárolt-e.

A PRIMO az ismertetett elvnek megfelelően rögzíti a jeleket a magnóra. Minden bit tárolására egy szimmetrikus négyszögimpulzus szolgál. Ennek periódusideje a rögzített bit értékének függvényében különböző. Az alábbi két ábra a 0, ill. 1 értékű biteknek megfelelő négyszögimpulzus alakját mutatja be:

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

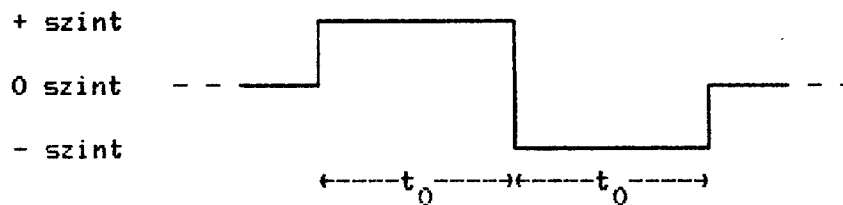
1 értékű bit jelalakja

$t_1 = 312$ mikrosec



0 értékű bit jelalakja

$t_0 = 936$ mikrosec



Amint az ábrákból látható, a négyszögimpulzusok a földpotenciálhoz képest pozitív, ill. negatív feszültségszintekből épülnek fel. Ezzel a három állapotú megoldással elkerülhető a magnóra kerülő nyugalmi és rögzítéskori egyenáramú komponens változása, amely a jelsorozat első és utolsó bitjének torzulását eredményezné. A szimmetrikus felépítés eredményeként az alkalmazott magnó esetleges polaritásfordításának - amely a visszaolvasott jel pozitív és negatív feszültségszintjeinek felcserélésével jelentkezik - hatása mérsékelhető.

A 0, ill. 1 értékeket ábrázoló négyszögimpulzusok arányának 1:3 értéket választottunk. E viszonylag nagy arány segítségével a magnó sebességingadozásának és torzításának jelmódosító hatása ellenére a bit értéke még biztonságosan visszaolvasható.

Mivel minden adatbitet hordozó hullámalak az előzőekben jellemzett tulajdonságokkal bír, elegendő egyetlen hullámalak detektálásának mikéntjét bemutatni.

Egy bit visszaolvasása a magnóról

Az egy bit visszaolvasását végző rutin a magnóról érkező - a PRIMO megfelelő áramkörei által digitalizált - hullámalak jellemző időtartamainak mérésével állapítja meg, hogy az adott bit 0, ill. 1 értékű-e. A bit azonosításának menete a következő:

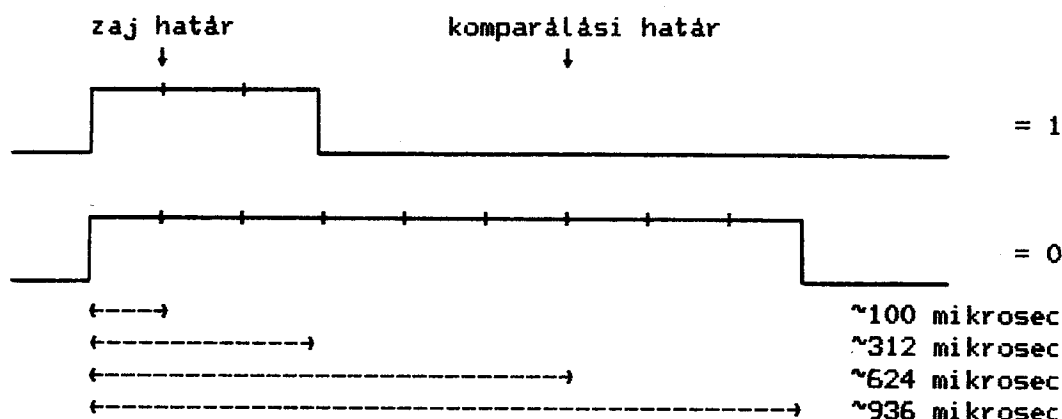
1. A rutin addig várakozik, amíg a magnóról érkező digitalizált jel értéke "+" szintű - logikai "1" - nem lesz.
2. Miután a jel "+" szintű lett, megkezdődik a "+" szint időtartamának mérése. Ennek során a bit beolvasó rutin folyama-

----- A PRIMO SZAMITOGEP MOKODESE -----

tosan mintavételezi a magnó bemenetet mindaddig, amíg a jel "-" vagy "0" szintű - logikai "0" - nem lesz. Eközben a mintavételek számát egy regiszterben őrzi. A mintavételezés befejeztével a számláló értékének vizsgálata következik.

3. Mivel a magnóról gyakran rövid idejű zajimpulzusok is érkeznek, a rutin elsőként azt vizsgálja, hogy a beolvasott jel "+" szintű tartományának ideje eléri-e az 1 értékű bit hasonló részének - az u.n. alapidőnek (312 mikrosec) - 1/3-át. Ha a vizsgált impulzus e határértéknél rövidebb, a PRIMO a jelet zajnak tekinti, és működését az 1. ponttól folytatja.
4. Ha a jel ~100 mikrosec-nál tovább tart, akkor a rutin azt vizsgálja, hogy az alapidő kétszeresénél is hosszabb-e. Ha hosszabb, akkor a beolvasott bit értékét 0-nak tekinti, míg rövidebb jel esetén a bitet 1-nek értelmezi.

Az ismertetett eljárással $\pm 30\%$ bit-idő ingadozás még nem befolyásolja az azonosítás menetét. Ez jelentősen növeli a beolvasás biztonságát, mivel a digitalizálás után mérhető bit-időt egyrészt a magnó sebességingadozása, másrészt torzítása módosíthatja. Az eddig leírtakat a következő ábra szemlélteti:



A bit beolvasás folyamatának áttekintése után a következőkben az állomány-szinkron mező feladatát és felépítését ismertetjük.

Állomány-szinkron mező

A visszaolvasás biztonságát lényegében a magnó sebességingadozása, torzítása, valamint a szalag minősége határozza meg. A jelfeldolgozást az is befolyásolja, hogy az adott magnó megváltoztatja-e a jel polaritását. E tényezők hatásának csökkentése érdekében a PRIMO minden állomány rögzítésekor a szalagra egy állomány-szinkron mezőt ír fel. E mező visszaolvasáskor történő elemzése segítségével a hibák jelentős része kiküszöbölhető.

----- A PRIMO SZÁMÍTÓGÉP MŰKÖDÉSE -----

Az állomány-szinkron mező 512 db 170;AA értékű byte-ból áll. Egy állomány betöltésekor a PRIMO e bitsorozat vizsgálatával a magnó sebességére és polaritásviszonyára vonatkozó információkat szerezhet. Az állomány-szinkron mező feldolgozásának menete a következő:

1. A PRIMO megkeresi az állomány-szinkron mező elejét. A mező kezdete abból ismerhető fel, hogy az első 170;AA értékű byte előtt a szalagon legalább 20 msec időtartamú jelmentes rész található. Ha a PRIMO felismeri az állomány-szinkron mezőt, megkezdí annak feldolgozását.
2. A soronkövetkező 4096 bit egy 10101010... jelsorozatnak felel meg. E szimmetrikus jelsorozatban mért ~~rövid és hosszú~~ impulzusok idejének arányából a PRIMO megállapítja, hogy az adott magnó megváltoztatja-e a rögzített impulzusok polaritását. A megszerzett információt a feldolgozó program a RAM 16479;405F címén tárolja. (Ha a magnó polaritást fordít, egy bit beolvasásánál a PRIMO az alacsony és magas szinteket majd felcserélve értelmezi.)
3. A polaritásviszonyok elemzése után a PRIMO beolvassa a magnetofonról érkező 10101010... jelsorozat következő 256 bit hosszúságú szakaszát, és átlagolja az ebben található bitek periódusidejét. Az így képzett értékből megállapítható az adott magnó átlagsebessége, kiszámítható egy bit alapideje. A kapott értéket a feldolgozó program a RAM 16480;4060 címén tárolja. A módszer eredményeként a PRIMO ~50%-os abszolút sebesség-ingadozás kompenzálására képes.

Blokk-szinkron mező

Egy blokk beolvasását követően megtörténik a beérkezett információ feldolgozása. Ennek ideje alatt azonban a magnószalag - hacsak a távirányításról nem gondoskodunk - továbbhalad. Mivel a blokkok feldolgozásának pontos ideje általában nem állapítható meg - az érték blokkonként eltérő is lehet -, minden blokk feldolgozása után a PRIMO a blokk-szinkron mező segítségével működését a magnóról érkező jelekhez szinkronizálja. A blokk-szinkron mező 96 db 255;FF és az ezt követő 3 db 211;D3 byte-ból áll. Az ütemezés folyamata a következő:

1. A PRIMO beolvassa a blokk-szinkron mező soronkövetkező bitjét, majd annak értékét elhelyezi egy 8 bites regiszter legalacsonyabb helyiértékű bitjében.
2. Ezután a feldolgozó program beolvassa a következő bitet, és a regiszter korábbi tartalmának egy pozícióval történő balra léptetése után az új bitet elhelyezi a legalacsonyabb helyi-értéken.

----- A PRIMO SZÁMÍTÓGÉP MÓKODÉSE -----

3. Most a PRIMO megvizsgálja, hogy a regiszter aktuális értéke egyenlő-e 211;D3-al. Ha még nem, akkor a blokk-szinkron mező feldolgozását a 2. ponttól folytatja.
4. Amennyiben a regiszter értéke 211;D3-al egyenlő, a PRIMO megtalálta a bitsorozatban a byte-határt. A továbbiakban az információ logikai beolvasása már nem bitenként, hanem 8 bites byte-ok formájában folytatható.

3.9.3. Magnó kezelő szubrutinok

A magnó logikai és fizikai kezelésének áttekintése után megadjuk a korábban ismertetett feladatokat megvalósító szubrutinok belépési címeit, valamint híváskor megadandó paramétereit.

Név-blokk felírása - WRHEAD

Belépési cím: 00146;0092

A szubrutin bekapcsolja a magnó távvezérlést (a PRIMO hátlapján található "TAPE" jelű csatlakozó 5-ös pontján +5V feszültség jelenik meg). Ezután 4 másodpercig várakozik. Ennek eredményeként a szalagon az előző és a most rögzítendő állomány között egy könnyen felismerhető "csendet" tartalmazó szalagrész alakul ki. A 4 másodperc letelte után felírja a szalagra az állomány-szinkron mezőt, egy blokk-szinkron mezőt, majd végezetül az állomány nevét tartalmazó név-blokkot.

Bemeneti paraméterek:

(HL) = állománynév-leíró címe
(16476;405C) = név-blokk típusa — 11.57.5.

Kimeneti paraméterek:

(16478;405E) = 0, kezdeti blokk-sorszám

Adat-blokk felírása - WRREC

Belépési cím: 00149;0095

A szubrutin a bemeneti paraméterek által meghatározott memória területet - 1/4 Kbyte méretű blokkokra bontva - a magnóra rögzíti. A kiírt blokk(ok) betöltési-cím mezejébe az aktuális blokk 1. byte-jának (memória cím - eltolási-cím) értéke kerül.

----- A PRIMO SZÁMÍTÓGÉP MŰKÖDÉSE -----

Bemeneti paraméterek:

(BC) = eltolási-cím
(DE) = memória terület kezdőcíme
(HL) = memória terület végcíme
(16476;405C) = adat-blokk(ok) típusa
(16478;405E) = az első rögzítendő blokk sorszáma - 1

Kimeneti paraméterek:

(16478;405E) = az utolsó rögzített blokk sorszáma

Záró-blokk felírása - ENDREC

Belépési cím: 00155;009B

A rutin egy blokk-szinkron mező kiírása után egy szabványos felépítésű záró-blokkot rögzít a magnóra. Ezután kikapcsolja a magnó távvezérlést.

Bemeneti paraméterek:

(16476;405C) = a záró-blokk típusa
(16478;405E) = a záró-blokk sorszáma *(amit a 149-es hagy)*

Név-blokk keresése/beolvasása - RDHEAD Belépési cím: 00143;00BF

15545

→ A rutin bekapcsolja a magnó távvezérlést, majd a szalagon - minimum 20 msec időtartamú - "csendet" keres. Ennek felismerése után feldolgozza a soronkövetkező állomány-szinkron mezőt, átlépi a blokk-szinkron mezőt és beolvassa a név-blokkot. Ezután összehasonlítja a megadott és a megtalált állománynevet és típust. Ha azok eltérőek, a rutin működését a "csend" kereséstől folytatja (átlépi az éppen megtalált állományt). Ha a név és típus egyezik, a rutin visszatér a hívó programba. (Ha a megadott állománynév rövidebb, mint az éppen beolvasott név, de a hosszabb név első részének karakterei rendre azonosak a megadott név karaktereivel, a PRIMO a két nevet azonosnak tekinti. Ha nem adunk meg állománynevet - név-hossz = 0 - a rutin az első egyező típusú állomány megtalálásakor visszatér a hívó programba.)

Bemeneti paraméterek:

(HL) = állomány-név leíró címe
(16476;405C) = a keresett állomány típusa *- 11 - 57.0*

Kimeneti paraméterek:

(16477;405D) = 0, kezdeti hibás blokk-számláló *induló értéke*
(16479;405F) = Fázisfordítás állapotjelző
(16480;4060) = Számított bit idő

Blokk-szinkronizálás - RDSYN

Belépési cím: 15477;3C75

(A rutin kezdőcíme - mivel nem ugrótábla-cím - A PRIMO '84.1 verziójára érvényes.)

A rutin megkeresi a szalagon a soronkövetkező blokk-szinkron mezőt, végrehajtja a PRIMO szinkronizálását, majd visszatér a hívó programba.

Kimeneti paraméterek:

(A) = a soronkövetkező blokk típusa

Adat-blokk beolvasása - PRGRC

Belépési cím: 00152;009B

A rutin beolvassa a - tetszőleges típusú - soronkövetkező blokkot. A szubrutin hívása előtt végre kell hajtani a blokk szinkronizációt, és be kell olvasni a blokk-sorszámot. A beolvasott blokk 1. byte-ja a blokkban lévő betöltési-cím, és a megadott kezdőcím összege által kijelölt memória címre kerül. *Kiírja a blokk sorszámát és a hiba számot,*

Bemeneti paraméterek:

(BC) = memória terület kezdőcíme

Egy byte beolvasása - INBYTE

Belépési cím: 00158;009E

A szubrutin beolvassa a szalagon soronkövetkező byte-ot alkotó 8 bitet, majd visszatér a hívó programba.

Bemeneti paraméterek:

(D) = kezdeti/aktuális ellenőrző-összeg értéke

Kimeneti paraméterek:

(A) = a beolvasott byte

(D) = a beolvasott byte értékével módosított ellenőrző-összeg

3.10. Direkt kurzor címző rutin - DIRCUR

A DIRCUR rutin segítségével a kurzort a képernyőn a megadott sor/oszlop címre állíthatjuk.

A kívánt sor, ill. oszlop címeket a mikroprocesszor D, ill. E regiszterébe kell a DIRCUR rutin meghívása előtt elhelyezni. A lehetséges értékek a következők:

----- A PRIMO SZAMITOGEP MOKODASE -----

Vízszintes írás esetén: sor cím = 0...15
oszlop cím = 0...41

Függőleges írás esetén: sor cím = 0...20
oszlop cím = 0...31

Vízszintes írásnál a képernyő legfelső sora a 0-ás sor, a soron belül a balszélső pozíció a 0-ás oszlop. Függőleges írásnál a kép az óramutató járásával ellenkező irányban, 90 fokkal el van forgatva, így a 0-ás sor a képernyő balszélső sora, a soron belül a 0-ás oszlop a kép alján található.

Ha a sor, ill. oszlop cím a megengedett tartományon kívül esik, a kurzor pozíciója nem változik, és a vezérlést a hívó rutin hibajelzéssel - (CY) = 1 - kapja vissza.

Belépési cím: 00128;0080

Bemeneti paraméterek:

(D) = sor cím
(E) = oszlop cím

Kimeneti paraméterek:

(CY) = 1, ha a sor, vagy oszlop cím hibás *A elvesz többi nem*

3.11. Kurzor pozíciót megállapító rutin - KURPOZ

A KURPOZ rutin megállapítja a kurzor aktuális képernyő soron belüli pozícióját, majd az értéket a mikroprocesszor A regiszterében adja vissza a hívó rutinnak. Ha a kurzor nem szabványos pozícióban áll - vagyis egy karakter bal felső pontjának X,Y címét a Felhasználó módosította -, a KURPOZ által adott érték a kurzor tényleges tartózkodási helyéhez legközelebb eső, annál nagyobb szabványos karakterpozíció lesz.

Belépési cím: 00140;008C

Kimeneti paraméterek:

(A) = kurzor aktuális soron belüli szabványos pozíciója

3.12. Grafikus pont kezelő rutinok

A PRIMO képernyője vízszintesen 256, függőlegesen 192 pontot tartalmaz. A 49152 pont bármelyike a többitől függetlenül kivilágosítható, ill. elsötétíthető. Lehetőség van egy pont pillanatnyi állapotának lekérdezésére is. Az említett funkciókat a grafikus pont kezelő rutinok hajtják végre.

----- A PRIMO SZÁMITÓGÉP MŰKÖDÉSE -----

A képernyő bal alsó pontja a 0,0 koordinátájú, jobb felső pontja a 255,191 koordinátájú pont. A grafikus pont kezelő rutinok a koordinátákat működésük során az

$$X=x \qquad Y=191-y$$

képletek alapján konvertálják. Erre azért van szükség, mert különben a kép bal felső pontja lenne a 0,0 koordinátájú pont. (A grafikus pont kezelő rutinok nem támogatják a PRIMO - egyébként lehetséges - 256*256 pontos üzemmódját.)

Egy grafikus pontot akkor nevezünk aktívnek, ha színe a képernyő aktuális alapszínének ellentétje, vagyis fekete képernyő alapszín esetén fehér, ill. fehér alapszín esetén fekete.

Az egyes grafikus pont kezelő rutinok bemeneti paramétereit azonosan kell megadni. A rutin működése után visszaadott értékek is hasonlóak. A következőkben ismertetjük a paraméterek megadásának, és a visszkapott adatok értelmezésének egységes szabályait.

Bemeneti paraméterek:

- (D) = az érintett pont y koordinátája (0...191)
- (E) = az érintett pont x koordinátája (0...255)

Kimeneti paraméterek:

- (D) = az érintett pont konvertált - Y - koordinátája
- (E) = az érintett pont konvertált - X - koordinátája
- (CY) = 1, ha az y koordináta nagyobb, mint 191

Grafikus pont bekapcsolása - SETDOT Belépési cím: 00131;00B3

A rutin bekapcsolja - aktívra teszi - a bemeneti paraméterek által kiválasztott grafikus pontot.

Grafikus pont kikapcsolása - CLRDOT Belépési cím: 00134;00B6

A rutin kikapcsolja - inaktívra teszi - a bemeneti paraméterek által kiválasztott grafikus pontot.

Grafikus pont vizsgálata - TSTDOT Belépési cím: 00137;00B9

A rutin megvizsgálja a bemeneti paraméterek által kiválasztott grafikus pontot. Ha aktívra találja, a ZERO flag értékét 0-ra állítja be. Inaktív pont esetén a flag értéke 1 lesz.

FÜGGELÉK

A PRIMO BASIC hibajelzései	F-1
PRIMO karakterkódok	F-7
Vezérlő karakterek és értelmezésük	F-9
Billentyűk periféria címei	F-15
Felhasználói karakterek definiálása	F-17
A PRIMO hanggenerátor paraméterezése	F-23
A PRIMO memória felosztása	F-27
PRIMO rendszerváltozók	F-29
PRIMO BASIC Interpreter változók	F-33
A PRIMO Bemeneti/Kimeneti kapui (I/O Portok)	F-39
PRIMO BASIC kulcsszavak belső kódjai	F-45
Konverziós táblázatok	F-49
Program listák	F-53

----- FELJEGYZÉSEK -----

A PRIMO BASIC hibajelzései

A következőkben felsoroljuk a PRIMO BASIC hibajelzéseit. A leírásban megadjuk a hiba két betűből álló emlékeztető kódját, az emlékeztető kódnak megfelelő angol nyelvű hibaszöveget, ennek magyar nyelvű fordítását, és az adott hiba belső kódját. A hibára jellemző belső kód ismerete azért szükséges, mert a programozott hibajelzés - ERROR utasítás - alkalmazásakor az ERROR kulcsszó után megadott kifejezés aktuális értékének ezt a kódot kell - a kívánt hibajelzés kiváltása érdekében - megadnia.

Ahogy azt a PRIMO BASIC leírásban említettük, valamely hiba felismerése után a PRIMO beállítja az ERR belső változó értékét. A különböző hibákhoz tartozó ERR-értékek a hiba belső kódjából a következő összefüggés segítségével határozhatók meg:

$$\text{ERR-érték} = 2 * (\text{belső kód} - 1)$$

- 1 NF NEXT without FOR FOR nélküli NEXT

Nincs olyan - eddig még le nem zárt - ciklus, amelynek ciklusváltozója megegyezne a NEXT utasításban megadott ciklusváltozóval, vagy - ha a NEXT után nem áll ciklusváltozó -, nincs egyetlen eddig még le nem zárt ciklus sem.

- 2 SN SyNtax Error Formai hiba

A PRIMO a BASIC program végrehajtása során egy olyan sort talált, amelyben formai hiba van. A hiba oka leggyakrabban helytelen zárójelezés, nem megengedett karakterek használata, vagy helytelen írásjelek alkalmazása. Formai hiba megjelenése után a PRIMO automatikusan megjeleníti a hibás sort, majd a következő képernyősor elején villogó kurzorral jelzi, hogy felkészült a sor szerkesztésére. Ezután a PRIMO BASIC nyelv leírásában ismertetett módon megkezdhető a hibás sor javítása.

- 3 RG RETURN without GOSUB GOSUB nélküli RETURN

A BASIC program végrehajtása alatt a PRIMO egy olyan RETURN utasítást talált, amelyet a programfutás során nem előzött meg GOSUB utasítás.

----- FOGGELÉK -----

4 OD Out of Data Kimerült a DATA-lista

Egy **READ** utasítás végrehajtásakor - miközben még van a **READ** utasításban értékre váró változó - a **PRIMO** elérte a **DATA**-lista végét.

5 FC Illegal Function Call Illegális funkcióhívás

Az **FC** hibajelzés megjelenése számos különféle hibaok valamelyikének bekövetkeztét jelzi. Általánosságban ez a hibajelzés akkor jelenik meg, ha egy parancs, utasítás vagy függvény operandusa nem megfelelő típusú és/vagy az értelmezési tartományon kívül esik. A hibajelzés okai részletesen a következők:

- **AUTO** parancs növekménye egyenlő nullával
- **DELETE** parancsban a törlendő programrész kezdősorszáma nagyobb, mint a végsorszám
- indexelt változó valamelyik indexe kisebb, mint nulla
- **CREATE**, **LOAD**, **OPEN** vagy **SAVE** utasításokban megadott program, ill. állománynév hosszabb, mint 16 karakter
- **PRINT#** utasítás kifejezés listájában **TAB** függvényt alkalmaztunk
- **PRINT USING** utasításban a formátumvezérlő karaktersorozat egyetlen karaktert sem tartalmaz (hossza = 0)
- **PRINT USING** utasításban a formátumvezérlő karaktersorozatban egy számspecifikáció az egész- és törtjegyek együttes számát 24 számjegynél többnek jelöli ki
- **VARPTR** függvény argumentumaként olyan változót adtunk meg, amely eddig még nem jelent meg egy kifejezésben, vagy értékadásban
- **CLEAR** utasítás operandusának aktuális értéke kisebb, mint nulla
- **ERROR** utasításban a megadott hibakód nem az 1...255 tartományba esik
- **ON x GOTO/GOSUB** utasításban **x** kifejezés aktuális értéke nem a 0...255 tartományba esik
- **ASC** vagy **STRING\$** függvény karakteres argumentumaként megadott karaktersorozat egyetlen karaktert sem tartalmaz
- **CHR\$, INP, LEFT\$, MID\$, RIGHT\$** vagy **STRING\$** függvények numerikus operandusának aktuális értéke nem a 0...255 tartományba esik
- **OUT** és **POKE** utasításban megadott adatbyte nem esik a 0...255 tartományba
- a **POINT** függvényben vagy a **RESET** és **SET** utasításban megadott **x**-, vagy **y**-koordináta kívül esik a megengedett tartományon
- **PRINT\$** utasításban a megadott sor-, vagy oszlopcím nem esik a megengedett tartományba

- TAB függvény argumentuma nagyobb, mint 41
- LOG függvény argumentuma kisebb, vagy egyenlő nullával
- RND vagy SQR függvény argumentuma kisebb, mint nulla

6 OV Overflow Túlcsordulás

Valamely aritmetikai művelet eredménye a **PRIMO BASIC** aktuális számbázis tartományában nem ábrázolható. A megengedett tartomány egész típusú számbázis esetén -32768...32767, egyszeres- és duplapontos valós számok alkalmazásakor pedig $\pm 1.70141E+38$.

(Ha a műveletvégzés során alácsordulás következik be, akkor a **PRIMO** az eredményhez zérus értéket rendel és a program végrehajtását hibajelzés nélkül folytatja.)

7 OM Out of Memory Betelt a memória

A rendelkezésre álló memória nem elegendő a **BASIC** program számára. Ennek okai a következők lehetnek:

- túl nagy a **BASIC** program
- túl sok skalár változót, és túlságosan nagy méretű tömböket alkalmaztunk
- túl sok **FOR** ciklust és szubrutinhívást ágyaztunk egymásba
- túlságosan bonyolult aritmetikai kifejezést alkalmaztunk

8 UL Undefined Line Nemdefiniált sor

A **GOSUB**, **GOTO**, **IF-THEN-ELSE**, **ON ERROR GOTO**, **ON GOSUB**, **ON GOTO**, **RESTORE**, vagy **RESUME** utasításban, vagy az **EDIT** és **RUN** parancsban megadott sorszámú sor nem található a memóriában tárolt **BASIC** programban.

9 BS Bad Subscript Hibás index

Egy tömbváltozó valamelyik indexének aktuális értéke meghaladja a tömb dimenzionálásakor meghatározott maximális értéket, vagy egy tömb dimenzionálásakor olyan nagy indexértéket adtunk meg, hogy a tömb tárolására nincs már elegendő memóriahely, vagy kevesebb, ill. több indexet adtunk meg, mint a tömb dimenzionálásakor.

10 DD Double Dimensioned Tömb újradimenzionálás

Egy korábban már dimenzionált tömbváltozót kíséreltünk meg ismét dimenzionálni, vagy egy korábban még nem dimenzionált, de már felhasznált tömböt akartunk újradimenzionálni. (Ez utóbbi eset azért vált ki hibajelzést,

----- FOGGELÉK -----

mert ha a **PRIMO** egy még nem dimenzionált tömböt észlel, úgy azt automatikusan dimenzionálja, tehát a későbbiekben ez a tömbnév már nem szerepelhet a **DIM** utasításban.)

11 /0 Division by Zero Nullával osztás

Az osztás műveletben az osztó aktuális értéke nulla, vagy hatványozásnál negatív hatványkitevő mellett az alap aktuális értéke nulla.

12 ID Illegal Direct Direkt módban nem alkalmazható

INPUT vagy **INPUT#** utasítást direkt módban próbáltunk alkalmazni. Az **INPUT** utasítás mindkét fajtája csak programból hajtható végre.

13 TM Type Mismatch Tipuskeveredés

Egy karakteres változóhoz numerikus értéket, vagy egy numerikus változóhoz karakteres értéket próbáltunk rendelni, ill. egy numerikus függvényargumentum helyén karakteres értéket szolgáltató kifejezés, vagy egy karakteres argumentum helyén numerikus értéket adó kifejezés áll.

14 OS Out of String Space Betelt a karakterterület

A **PRIMO** karaktersorozatok tárolására elkülönített memóriaterülete betelt. A hibát a karakterterület méretének megnövelésével - **CLEAR** utasítás - küszöbölhetjük ki.

15 LS Long String Hosszú karaktersorozat

Karaktersorozatok összefűzése - konkatenáció - során 255 karakternél hosszabb karaktersorozat keletkezett.

16 ST String Formula Too Complex Túlságosan bonyolult karakteres kifejezés

A karaktersorozatokot tartalmazó kifejezés túlságosan bonyolult a **PRIMO** számára. A kifejezést két vagy több részre bontva kell a **BASIC** programban megadni.

17 CN Can't Continue Nem folytatható

A **CONT** paranccsal megkíséreltük a **BASIC** program továbbindítását, de:

- a memóriában nincs **BASIC** program
- a **BASIC** program hibajelzéssel állt le

- a program leállítása és CONT paranccsal megkísérelt újraindítása között eltelt időben módosítottuk a memóriában tárolt BASIC programot.

- | | | | |
|--|----|----------------------|------------------------|
| 18 | NR | No RESUME | Lezáratlan hibakezelés |
| <p>Miközben egy programozott hibakezelő rutin végrehajtása folyik, a PRIMO elérte a program utolsó sorát azelőtt, hogy egy RESUME utasítás előfordult volna.</p> | | | |
| 19 | RW | RESUME Without Error | Hiba nélküli RESUME |
| <p>A PRIMO programozott hibakezelő rutinon kívül egy RESUME utasítást talált a BASIC programban.</p> | | | |
| 20 | UE | User Error | Felhasználói hiba |
| <p>A PRIMO olyan hibát észlelt, amelyhez nem tartozik szabványos hibaüzenet. Ezt a hibajelzést olyan ERROR utasítás végrehajtása eredményezi, amelyben a megadott hibakód nagyobb, mint 23.</p> | | | |
| 21 | MD | Missing Operand | Hiányzó operandus |
| <p>Egy kifejezésben a műveleti jelet nem követi operandus.</p> | | | |
| 22 | FD | File Description | Allományleírás hiba |
| <p>FD hibajelzés a magnetofonos adatkezelés alkalmazása során keletkezhet. A hibajelzést kiváltó okok a következők:</p> <ul style="list-style-type: none"> - A BASIC programban CLOSE utasítás következik, miközben nincs megnyitott adatállomány - PRINT# utasítást alkalmaztunk anélkül, hogy CREATE utasítással az adatállományt megnyitottuk volna - INPUT# utasítást alkalmaztunk úgy, hogy az adatállományt korábban OPEN utasítással nem nyitottuk meg | | | |

Bármely, az előzőekben fel nem sorolt hibakód megjelenése egy ERROR utasításban az UE hibajelzés megjelenését eredményezi.

----- FOGSÉLÉK -----

A következőkben névsorba rendezve felsoroljuk a PRIMO BASIC hibajelzések emlékeztető kódjait, a hozzájuk rendelt belső hibakódot és az emlékeztető kódnak megfelelő hibaszövegeket.

BS	9	Bad Subscript	Hibás index
CN	17	Can't Continue	Nem folytatható
DD	10	Double Dimensioned	Tömb újradimenzionálás
FC	5	Illegal Function Call	Illegális funkcióhívás
FD	22	File Description	Állományleírás hiba
ID	12	Illegal Direct	Direkt módban nem alkalmazható
LS	15	Long String	Hosszú karaktersorozat
MO	21	Missing Operand	Hiányzó operandus
NF	1	NEXT without FOR	FOR nélküli NEXT
NR	18	No RESUME	Lezáratlan hibakezelés
OD	4	Out of Data	Kimerült a DATA-lista
OM	7	Out of Memory	Betelt a memória
OS	14	Out of String Space	Betelt a karakterterület
OV	6	Overflow	Túlcsordulás
RG	3	RETURN without GOSUB	GOSUB nélküli RETURN
RW	19	RESUME without ERROR	Hiba nélküli RESUME
SN	2	Syntax Error	Formai hiba
ST	16	String Formula Too Complex	Túlságosan bonyolult karakteres kifejezés
TM	13	Type Mismatch	Típuskeveredés
UE	20	User Error	Felhasználói hiba
UL	8	Undefined Line	Nemdefiniált sor
/O	11	Division by Zero	Nullával osztás

PRIMO karakterkódok

A **PRIMO** a 7 bites **ASCII** - American Standard Code for Information Interchange - kódrendszer szabványos svéd kiterjesztéséből kialakított kódrendszert alkalmazza. Ennek megfelelően az ékezetes betűk egy része - á Ä é É ö Ö ú Ú - a kódrendszerben úgy helyezkedik el, ahogy azt az **ASCII** svéd kiterjesztése előírja, míg a magyar **ABC** hosszú ékezetes betűi néhány - a **PRIMO** által nem használt - speciális karakter - @ [\] + (|) ~ - helyére kerültek. Ezzel a megoldással elértük, hogy a csak rövid ékezetes karaktereket tartalmazó szöveg mindazon nyomtatóval helyesen megjeleníthető, amely ismeri az **ASCII** kódrendszer svéd kiterjesztését.

A **PRIMO** néhány különleges karakter megjelenítésére is képes. Ezek a karakterek a Felhasználó által definiálható karakterkódok egyikén helyezkednek el, így új karakterek definiálása után már nem használhatók.

A következőkben két táblázatba rendezve felsoroljuk a **PRIMO** valamennyi megjeleníthető karakterét. Az első táblázatban a karakterek decimális kódjuk, a másodikban hexadecimális kódjuk szerint rendezettek. (A vezérlő karakterek kódjai és értelmezése az F-9 oldalon kezdődő fejezetben található.)

Megjeleníthető karakterek decimális kódjai

	30	40	50	60	70	80	90	100	110	120	130	140	150
0	i	(	2	(	F	P	Z	d	n	x	]	§	Σ
1	†	)	3	=	G	Q	ó	e	o	y	+	≠	Π
2		*	4	)	H	R	Ö	f	p	z	→	@	
3	!	+	5	?	I	S	A	g	q	ő	↓	(	
4	"	,	6	é	J	T	U	h	r	ö	√	)	
5	#	-	7	Ä	K	U	ú	i	s	ä	✓	~	
6	\$	.	8	B	L	V	é	j	t	ü	√	Ω	
7	z	/	9	C	M	W	a	k	u	û	Ft	μ	
8	&	0	:	D	N	X	b	l	v	π	l	2	
9	'	1	;	E	O	Y	c	m	w	[	~	∞	

A 128...151 kódú karakterek csak akkor alkalmazhatók, ha nem definiáltunk új karaktereket.

Újonnan definiált karakterek kódja a 128...255 tartományba eshet.

Megjeleníthető karakterek hexadecimális kódjai

	10	20	30	40	50	60	70	80	90
0			0	é	P	é	p	π	)
1		!	1	A	Q	a	q	[	~
2		"	2	B	R	b	r	]	Ω
3		#	3	C	S	c	s	+	μ
4		\$	4	D	T	d	t	+	2
5		%	5	E	U	e	u	+	∞
6		&	6	F	V	f	v	√	Σ
7		'	7	G	W	g	w	√	Π
8		(	8	H	X	h	x	√	
9		)	9	I	Y	i	y	Ft	
A		*	:	J	Z	j	z	l	
B		+	;	K	ó	k	ö	~	
C		,	<	L	ó	l	ö	§	
D		-	=	M	A	m	á	≠	
E	i	.	>	N	ó	n	ü	@	
F	†	/	?	ó	á	o	ä	(	

A 80H...97H kódú karakterek csak akkor alkalmazhatók, ha nem definiáltunk új karaktereket.

Újonnan definiált karakterek kódja a 80H...OFFH tartományba eshet.

Vezérlő karakterek és értelmezésük

A PRIMO kódrendszerében a 1...25 decimális - 1...19H hexadecimális - kódú karakterek vezérlő karakterek. E kódokhoz nincs a képernyőn megjelenő karakter rendelve, alkalmazásuk esetén csak a végrehajtott funkció eredménye látható.

A következőkben felsoroljuk a PRIMO vezérlő karaktereinek decimális és hexadecimális kódjait, az egyes kódok jelentését, valamint azt a billentyűt/billentyűket, amely/amelyek megnyomása az adott vezérlő kódot eredményezi.

dec hex billentyű funkció

01 01 ----- Normál

Hatására a képernyő kezelő program a továbbiakban normál üzemmódban működik tovább. Ennek megfelelően vízszintesen, fekete háttérre fehér, normál méretű betűket fog előtörölés nélkül megjeleníteni. (Ezt a kódot nem lehet a billentyűzetről begépelni. A BASIC programban a PRINT CHR\$(1) utasítással váltható ki.)

02 02 CTR/B Nyújtott méretű karakter megjelenítés

Megadása után a képernyőre kiírt karakterek vízszintes irányú mérete a normál 6 képpont helyett 12 képpont lesz. Ennek megfelelően a képernyő egy sorában 42 karakter helyett 21 nyújtott karakter írható ki. A képernyőn már látható karakterek nem változnak meg.

03 03 CTR/C Fehér képernyő alapszín beállítása

Alkalmazásának hatására a képernyő alapszíne fehér lesz. E kód megadása előtt a képernyőre kiírt karakterek invertáltan továbbra is láthatók.

04 04 CTR/D Inverz karakteralap beállítása

Megadása után a képernyőre kiírt karakterek a képernyő aktuális alapszínéhez képest invertált karakteralapon jelennek meg, vagyis fekete alapszín esetén fehér karakteralapon fekete jelek, fehér alapszín esetén fekete karakteralapon fehér jelek láthatók. A képernyőn már látható karakterek nem változnak meg.

05 05 CTR/E **Karakter aláhúzás bekapcsolása**

Megadása után, a képernyőre kiírt karakterek aláhúzva fognak megjelenni. A képernyőn már látható karakterek nem változnak meg.

06 06 CTR/F **Előtörles bekapcsolása**

A vezérlő kód megadása után a PRIMO egy karakter megjelenítését megelőzően a karakteralap 6*12 - nyújtott karakter esetén 12*12 - képpontját a képernyő aktuális alapszínével megegyező színűre állítja és csak ezután jeleníti meg a karakteralapon a kívánt betűt, számot vagy írásjelet. Az előtörles alkalmazásával elérhető, hogy a képernyő valamely pozíciójában az ott korábban megjelenített karaktert egy új jellel felülírjuk.

07 07 CTR/G **Hangjelzés**

A 7-es vezérlő karakter hatására a PRIMO hangszórója egy 1/6 másodperc hosszúságú, 800 Hz frekvenciájú hangot ad.

08 08 + **Utolsó karakter törlése**

A vezérlő kód alkalmazása eredményeként a PRIMO a kurzor aktuális pozíciójától balra álló karaktert törli a képernyőről, majd a kurzort a letörölt karakter helyére állítja. Ha a kurzor egy sor legelső pozíciójában volt, akkor az előző sor utolsó karaktere törlődik. Amennyiben a kurzor a legelső - legfelső - sor első pozíciójában áll, a vezérlő kód hatástalan.

09 09 SHIFT/+ **Vízszintes tabulátor**

Hatására a kurzor a sor következő, nyolccal maradék nélkül osztható oszlopszámú pozíciójára lép előre. Ha az adott sorban nincs már ilyen pozíció, a kurzor a következő sor első pozíciójába kerül. (Ez a vezérlő kód egy BASIC program-, ill. adatsor begépelésekor nem alkalmazható. A PRINT utasításban azonban CHR\$(9) alakban megadható.)

10 0A ↓ **BASIC EDITOR újraindítása**

Ez a vezérlő kód csak a BASIC EDIT parancsának alkalmazásakor használható. Hatásának részletes leírása a PRIMO BASIC nyelv leírásában található.

12 OC CLS **Képernyő törlés**

Megadása után a képernyő tartalma törlődik és valamennyi képpont színe a képernyő aktuális alapszínével lesz azonos. A vezérlő kód a kurzort a legfelső sor első pozíciójába állítja és kikapcsolja az előtörlést.

13 OD RETURN **Kocsi vissza + soremelés**

Hatására a kurzor a következő sor első pozíciójába áll. Ha a kurzor a képernyő legalsó sorában volt, akkor a kép utolsó 15 sora egy sorral feljebb lép - vagyis a legelső sor tartalma eltűnik az ernyőről -, és a képernyő legalsó sorában egy üres sor jelenik meg. A kurzor ezen üres sor első pozíciójába kerül.

14 OE CTR/N **Kurzor a sor elejére**

Alkalmazása esetén a kurzor azon sor első pozíciójába lép, amely sorban éppen tartózkodott.

15 OF CTR/O **Függőleges írás**

Megadása után a PRIMO képernyőkezelő programja a megjelenítendő karaktereket az óramutató járásával ellenkező irányban 90 fokkal elforgatva, függőlegesen írja ki. A képernyőn már látható karakterek nem változnak meg.

Függőleges írás esetén a képernyőn 21 sor, soronként 32 karakter jeleníthető meg. Ebben az üzemmódban hatástalan az "Utolsó karakter törlése", "Vízszintes tabulátor", "Kocsi vissza + soremelés" és "Kurzor a sor elejére" vezérlő kód. Függőleges írás üzemmódban a kurzor a sor végéről nem a következő, hanem az aktuális sor elejére lép. Így egy sornál - 32 karakter - hosszabb szöveg megjelenítése értelmesen nem lehetséges.

Függőleges írás esetén is alkalmazható a direkt kurzor címzés - PRINTS utasítás -. Ilyenkor azonban a megadható sor-, és oszlop címek a megváltozott képméret miatt a vízszintes írásnál megszokottaktól eltérőek. A lehetséges sor címek a 0...20, az oszlop címek a 0...31 tartományba esnek. A 0-ás című sor a kép bal szélén helyezkedik el, a 0-ás című oszlop a kép alján.

16 10 CTR/P **Alsó index**

Megadása után a képernyőre kiírt karakterek 4 képponttal - 1/2 karakterméret - lejjebb csúsztatva jelennek meg. A vezérlő kód ismételt alkalmazása újabb 4 képpontos lefe-

lé csúsztatást eredményez. Ha az alsó indexbe írt karakterek egy része már túlnyúlik a képernyő alsó szélén, a karakterek e túlnyúló részét a **PRIMO** nem jeleníti meg. A képernyőn már látható karakterek nem változnak meg.

17 11 CTR/Q Felső index

Megadása után a képernyőre kiírt karakterek 4 képponttal - 1/2 karakterméret - feljebb csúsztatva jelennek meg. A vezérlő kód ismételt alkalmazása újabb 4 képpontos felfelé csúsztatást eredményez. Ha a felső indexbe írt karakterek egy része már túlnyúlik a képernyő felső szélén, a karakterek e túlnyúló részét a **PRIMO** nem jeleníti meg. A képernyőn már látható karakterek nem változnak meg.

18 12 CTR/R Normál méretű karakter megjelenítés

Ez a vezérlő kód a "Nyújtott méretű karakter megjelenítés" kód hatását szünteti meg. Alkalmazása után a képernyőre kiírt karakterek vízszintes irányú mérete a normális 6 képpont lesz. A képernyőn már látható karakterek nem változnak meg.

19 13 CTR/S Fekete képernyő alapszin beállítása

Ez a vezérlő kód a "Fehér képernyő alapszin beállítása" kód hatását szünteti meg. Alkalmazása után a képernyő alapszine fekete lesz. E kód megadása előtt a képernyőre kiírt karakterek invertáltan továbbra is láthatók.

20 14 CTR/T Inverz karakteralap kikapcsolása

Ez a vezérlő kód az "Inverz karakteralap beállítása" kód hatását szünteti meg. Alkalmazása után a karakterek a képernyő aktuális alapszínének megfelelő karakteralapon jelennek meg. A képernyőn már látható karakterek nem változnak meg.

21 15 CTR/U Karakter aláhúzás kikapcsolása

Ez a vezérlő kód a "Karakter aláhúzás bekapcsolása" kód hatását szünteti meg. Alkalmazása után a képernyőre kiírt karakterek aláhúzás nélkül jelennek meg. A képernyőn már látható karakterek nem változnak meg.

22 16 CTR/V Előtörles kikapcsolása

Ez a vezérlő kód az "Előtörles bekapcsolása" kód hatását szünteti meg. Alkalmazása után a **PRIMO** egy karakter

megjelenítésekor nem törli a karakteralapot, hanem a kiírandó betűt, számot vagy írásjelet a képző korábbi tartalmára ráírja. Így elérhető, hogy a képernyő valamely pozíciójában két karakter egymásra írásával egy új jelet hozzunk létre. A **PRIMO** alaphelyzetben - és a "Képernyő törlés" kód megadása után - kikapcsolt előtöréssel üzemel.

23 17 CTR/W Vizszintes írás

Ez a vezérlő kód a "Függőleges írás" kód hatását szünteti meg. Alkalmazása után a **PRIMO** ismét vízszintesen írja ki a megadott karaktereket. A függőleges megjelenítés során nem értelmezett vezérlő kódok ismét működésbe hozhatók. A képernyőn már látható karakterek nem változnak meg.

24 18 SHIFT/+ Sorszerkesztés újrakezdése

Ez a vezérlő kód csak a **BASIC EDIT** parancsának alkalmazásakor használható. Hatásának részletes leírása a **PRIMO BASIC** nyelv leírásában található.

25 19 SHIFT/+ Sorszerkesztés lezárása

Ez a vezérlő kód csak a **BASIC EDIT** parancsának alkalmazásakor használható. Hatásának részletes leírása a **PRIMO BASIC** nyelv leírásában található.

----- FELJEGYZÉSEK -----

Billentyűk periféria címei

A PRIMO számítógép a billentyűzet egyes gombjait egy-egy perifériaként kezeli, így mindegyik gomb saját periféria címmel rendelkezik. A következő két táblázatban a periféria címek növekvő sorrendjében, ill. a gombok által képviselt karakterek ABC sorrendjében megadjuk az egyes gombok decimális és hexadecimális periféria címeit.

A periféria címek ismeretében egy gomb állapota akár BASIC, akár gépi kódú program segítségével ellenőrizhető. Ha a gombot BASIC programmal vizsgáljuk, az INP függvényt alkalmazhatjuk.

Példa: 100 IF INP(53) AND 1=1 THEN PRINT "CLS aktiv"

A példában az INP függvény értékét azért kell logikai ÉS művelet segítségével maszkolni, mert az INP függvény által szolgáltatott 8 bites értéknek csak a legkisebb helyiértékű bitje jelképezi a gomb pillanatnyi állapotát, míg a többi bit más jelentéssel bír.

Billentyű címek periféria cím szerint rendezve

dec	hex	gomb	dec	hex	gomb	dec	hex	gomb
00	00	Y	23	17	7 /	43	2B	. :
01	01	↑	24	18	H	44	2C	M
02	02	S	25	19	Betűköz	45	2D	9 ;
03	03	SHIFT	26	1A	B	46	2E	I
04	04	E	27	1B	6 &	47	2F	, ,
05	05	UPPER	28	1C	G	48	30	0
06	06	W	29	1D	5 %	49	31	' *
07	07	CTR	30	1E	V	50	32	P
08	08	D	31	1F	4 \$	51	33	ú û
09	09	3 #	32	20	N	52	34	0
10	0A	X	33	21	8 (	53	35	CLS
11	0B	2 "	34	22	Z	55	37	RETURN
12	0C	Q	35	23	+ ?	57	39	←
13	0D	1 !	36	24	U	58	3A	é
14	0E	A	37	25	0 =	59	3B	ó ô
15	0F	↓	38	26	J	60	3C	A
16	10	C	39	27	) <	61	3D	→
18	12	F	40	28	L	62	3E	0
20	14	R	41	29	- i	63	3F	BRK
22	16	T	42	2A	K			

----- FOGGELÉK -----

Billentyű címek a karakterkódok sorrendjébe rendezve

gomb	dec	hex	gomb	dec	hex	gomb	dec	hex
. :	43	2B	C	16	10	T	22	16
←	57	39	D	08	08	U	36	24
+ ?	35	23	E	04	04	0	48	30
↑	01	01	É	58	3A	ú û	51	33
↓	15	0F	F	18	12	V	30	1E
- i	41	29	G	28	1C	W	06	06
→	61	3D	H	24	18	X	10	0A
, ;	47	2F	I	46	2E	Y	00	00
> <	39	27	J	38	26	Z	34	22
' *	49	31	K	42	2A	0 =	37	25
Betűköz	25	19	L	40	28	1 !	13	0D
BRK	63	3F	M	44	2C	2 "	11	0B
CLS	53	35	N	32	20	3 #	09	09
CTR	07	07	O	52	34	4 \$	31	1F
RETURN	55	37	ó ô	59	3B	5 %	29	1D
SHIFT	03	03	ö	62	3E	6 &	27	1B
UPPER	05	05	P	50	32	7 /	23	17
A	14	0E	Q	12	0C	8 (	33	21
A	60	3C	R	20	14	9)	45	2D
B	26	1A	S	02	02			

A **PRIMO** billentyűzetén a következő periféria címekhez nincs gomb kapcsolva:

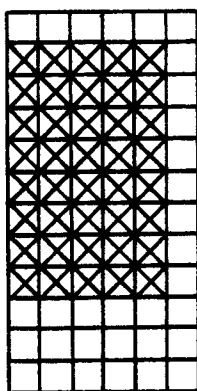
decimális 17, 19, 21, 54, 56
hexadecimális 11, 13, 15, 36, 38

Felhasználói karakterek definiálása

A PRIMO lehetővé teszi, hogy a Felhasználó új, a gép által még ismeretlen karaktereket hozzon létre, melyek definiálásuk után a szabványos jelekhez hasonlóan alkalmazhatók. A következőkben részletesen ismertetjük a karakterek definiálásához szükséges alapfogalmakat.

A Felhasználó által definiált karakterek kódjai a 128...255 intervallumba eshetnek. Így egy időben maximálisan 128 új karakter alkalmazható. Amint az a PRIMO karakterkódjait ismertető Függelékből is látható, a 128...151 kódok definiáltak, azok alkalmazása esetén a Függelékben található karakterek jelennek meg a képernyőn. (Ha a 152...255 kódok valamelyikét még definiálása előtt alkalmazzuk, a képernyőn a következő karakter helyén egy értelmetlen alakzat fog megjelenni.) Ha kihasználjuk a karakter definiálás lehetőségét, az említett 24 karaktert vagy újra definiáljuk, vagy alkalmazásukról le kell mondanunk. Ennek az az oka, hogy Felhasználói karakterdefiníció esetén a 128...151 kódokon korábban elhelyezkedő karaktereket a PRIMO "elfelejti".

A PRIMO normál méretű karakterei vízszintesen 5, függőlegesen 8 képpont kiterjedésűek. Az 5*8 pont egy 6*12 pontból álló mezőben helyezkedik el. Mivel a PRIMO képernyője 256*192 képpont kiterjedésű, így egy időben 16 sorban, soronként 42 karakter jeleníthető meg. Egy karakter a rendelkezésére álló mezőben a következő módon helyezkedik el:



Az egymás mellett álló karakterek elválasztását a baloldali karakter 6. pontoszlopa biztosítja, amely általában nem tartalmaz aktív pontokat.

A sorok elválasztását az általában aktív pontokat nem tartalmazó 1. és 10-12. karaktersorok biztosítják.

Mivel az egyes karaktereket elválasztó pontok is aktivizálhatók, így lehetséges vízszintes, ill. függőleges irányban összefüggő alakzatok létrehozása is. Ilyenkor több, megfelelően meg szerkesztett karakter együttesen alkot egy alakzatot.

Amikor a képernyőkezelő program egy karaktert a képernyőre rajzol, kikeresi a PRIMO ROM-jában elhelyezkedő karaktergenerátor területről a megfelelő alakzat pontmintázatát. Ez egy karakterenként 8 byte-os terület, amely meghatározza, hogy mely pontokat kell kigyújtani, ill. eloltani az 5*8 pont közül annak érdekében, hogy a képernyőn a kívánt betű, szám vagy írásjel megjelenjen.

A pontmintázat minden byte-ja a karakter egy sorát határozza meg. Mint említettük, egy karaktermező 6 pont széles és 12 pont magas. Egy sor meghatározásához a byte 6 bitje szükséges tehát. Az 1. oszlopot a 6-os bit, a 2. oszlopot az 5-ös bit, míg az 5. oszlopot az 1-es bit jelképezi. (A byte bal szélső bitje a 7-es, jobb szélső bitje a 0-ás sorszámú.) A megmaradt két bit - 7-es és 0-ás - különleges célokat szolgál.

Ha a karakter 1. sorát meghatározó byte 7-es bitjében 1 áll, a karakter ún. süllyesztett betű. Ilyen pl. az y vagy a j. Az ilyen karaktereket a képernyőkezelő program a 6*12 pontból álló mezőben nem a 2. sortól, hanem a 4. sortól kezdve jeleníti meg. Ennek megfelelően a karakter legalsó sora a mező 11. sorába kerül. Ezzel a megoldással elérhető, hogy egy nyomtatott vagy írott szöveghez hasonlóan a süllyesztett karakterek egyes részei az írásvonal alá nyúlva jelenjenek meg, de ezek a karakterek sem igényelnek 8 byte-nál hosszabb pontmintázatot. A karakter további sorait meghatározó byte-ok 7-es bitjeinek értéke közömbös.

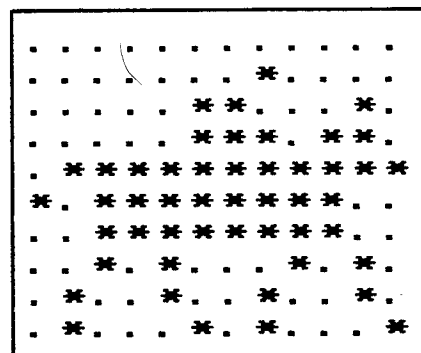
A 7-es bithez hasonlóan a 0-ás bitet is különlegesen értelmezi a képernyőkezelő program. Ezzel a bittel jelezzük a program számára, hogy kirajzolta már a karakter legalsó sorát, nem kell újabb sort meghatározó byte-ot olvasni a karaktergenerátor területről. Ez a megoldás azt az előnyt nyújtja, hogy 8 pontnál magasabb karaktereket is alkalmazhatunk. Így pl. több soron átnyúló integráljel is megjeleníthető. Ez úgy oldható meg, hogy a karakter nyolcadik sorát jelképező byte 0-ás bitjébe is 0-át írunk, így a program nem fejezi be a karakter megjelenítését, hanem további sorokat rajzol mindaddig, amíg az alakzat végét jelző bit értéke 1 nem lesz. Ekkor azonban a különleges karakter kódját követő kódot - kódokat - elveszítjük, hiszen az azok megjelenítéséhez szükséges 8 byte-os mezőt - mezőket - a különleges karakter pontmintázata részének tekinti a képernyőkezelő program.

A karaktergenerátor szervezése lehetővé teszi, hogy két egymás mellé írt jel összefüggő alakzatot alkosson. Ez annak köszönhető, hogy míg a pontmintázat hat pont széles, maga a karakter tipikusan csak a bal szélső öt pontot foglalja el. Ha ebben a 6. oszlopban is van a karakternek aktív pontja, nem lévén már elválasztó mező, a szomszédos karakterek egybefolynak. Az előzőekben leírtak figyelembe vételével vizsgáljuk meg pl. az A betű képét megadó 8 byte-os karaktergenerátor-mező felépítését és az egyes byte-ok decimális értékét:

byte	bit	7	6	5	4	3	2	1	0	decimális érték
1.		0	0	0	1	0	0	0	0	16
2.		0	0	1	1	1	0	0	0	56
3.		0	1	0	1	0	1	0	0	84
4.		0	1	0	0	0	1	0	0	68
5.		0	1	1	1	1	1	0	0	124
6.		0	1	0	0	0	1	0	0	68
7.		0	1	0	0	0	1	0	0	68
8.		0	1	0	0	0	1	0	1	69

A karaktergenerátor-mező felépítésének megismerése után vizsgáljuk meg, hogyan lehetséges az újonnan definiált karakterek alkalmazása. Ezt végső soron az teszi lehetővé, hogy a képernyőkezelő program a 128...255 kódú karakterek pontmintázatát jelképező karaktergenerátor-mező kezdőcímét a **PRIMO** rendszerváltozónak fenntartott **RAM**-területen őrzi. Így e mező kezdőcíme szabadon változtatható. Ha az ismertetett szabályok figyelembe vételével pl. a **BASIC** programterületen létrehozunk egy, az új karakterek pontmintázatát tartalmazó karaktergenerátor területet, és a rendszerváltozó terület megfelelő címére - 16459-16460;404B-404C - elhelyezzük ezen terület kezdőcímét, akkor a képernyőkezelő program a 128...255 kódú karakterek képét erről a helyről fogja megjelenítéskor kiolvasni.

Az eddig leírtak illusztrálására az alábbiakban végezzük el egy **Felhasználói** karakter megtervezését és definiálását. Példaként válasszunk egy olyan alakzatot, amely egy ló hátán ülő lovast ábrázol. A pontmintázat tervezésekor hamar kiderül, hogy kielégítő ábrát csak 6 pontnál szélesebb, 8 pontnál magasabb alakzat esetén kapunk. Az eddig leírtak alapján ez nem okoz problémát, mivel - több karakterkód felhasználásával - a **PRIMO** képes a szükséges méretű alakzat megjelenítésére is. Ezért nem egy, hanem két karaktert kell definiálnunk. Ezek közül az első az ábra bal felét, a második a jobb felét ábrázolja. Mindkét karaktert a szokásos 8 pontnál magasabbra tervezzük. A kitűzött célt megvalósító egy lehetséges alakzat - amely 12*10 pontból áll - a következő lehet:



----- FÖGGELEK -----

Az ábrának megfelelő karaktergenerátor-mező decimális kódjai a következők:

Bal oldal	byte	dec. kód	Jobb oldal	byte	dec. kód
1.		0	1.		0
2.		0	2.		32
3.		2	3.		68
4.		2	4.		108
5.		62	5.		126
6.		94	6.		120
7.		30	7.		120
8.		20	8.		20
9.		36	9.		36
10.		35	10.		35

Az ábra bal oldalát jelképező karakter kódjának válasszuk a 128-as kódot. Mivel a definiált ábra 8 pontnál magasabb, ez azt eredményezi, hogy a definícióval a 129-es kódot is elfoglaltuk. Ezért az ábra jobb oldalát a 130-as kód képviselheti, amely természetesen elfoglalja még a 131-es kódot is. Ezek után a teljes ábrát leíró karaktergenerátor-mező a következő:

128.129 kód = 0, 0, 2, 2, 62, 94, 30, 20, 36, 35, x, x, x, x, x, x
 130.131 kód = 0, 32, 68, 108, 126, 120, 120, 20, 36, 35

(A 128.129 kódhoz tartozó értéksor végén látható 6 db x a 129-es kódhoz tartozó karaktergenerátor-mező fel nem használt 6 byte-nyi részét jelképezi. E byte-ok megadása azért szükséges, hogy a képernyőkezelő program a 130-as kódú karakter elejét megtalálhassa, vagyis minden karakterkódhoz 8 byte-nyi területet kell definiálni még akkor is, ha azt ténylegesen nem is használjuk ki. Ez alól csak a legmagasabb kódú definiált karakter kivétel, amelynek nem használt sorait jelképező byte-ok megadása nem kötelező. A kihasználatlan - x-el jelölt - byte-ok értéke tetszőleges lehet.)

Az elkészített ábra programbeli definiálását és alkalmazását szemlélteti a következő oldalon látható bemutató program.

A program 120-as sorában álló DIM utasítással lefoglalt Kx vektor tárolja majd a definiált karakterek pontmintázatát. A C skalár változó a karaktergenerátor-mező kezdőcímét őrzi.

Ha az új karakterek pontmintázatát tároló karaktergenerátor-mezőt egy BASIC tömbváltozóban - jelen esetben a Kx vektorban - helyezzük el, akkor a program helyes működésének érdekében biztosítani kell, hogy a tömb kezdőcímének meghatározását végző VARPTR függvény alkalmazása után a programban korábban még nem használt skalár változó már ne szerepeljen. Egy ilyen változó megjelenése ugyanis azt eredményezi, hogy megváltozik a tömbváltozók tárolón

belüli kezdőcíme, tehát a képernyőkezelő program nem képes a karaktergenerátor-mezőt megtalálni. Ezt a problémát úgy szüntethetjük meg, hogy a program elején - mint a 110-es sorban is látható - egy fiktív értékadásban felsoroljuk a programban alkalmazott valamennyi skalár változót.

A karaktergenerátor-mező feltöltése a 140...145-ös sorban történik meg. A 150-es sorban lévő POKE utasítás állítja be a 128...255 kódú karakterek karaktergenerátor mezejének kezdőcímét tartalmazó rendszerváltozót a megfelelő értékre. (Ezzel egyidejűleg a 128...151 kódok alatt szereplő - a PRIMO által eredetileg ismert - karaktereket a gép "elfelejti" mindaddig, amíg a RESET gombot meg nem nyomjuk. Ekkor viszont az általunk definiált karakterek szűnnek meg a program újabb futtatásáig.)

```

100 'Löverseny DEMO '84.1
110 C=0 : Vx=0
120 DIM Kx(12) : C=VARPTR(Kx(0)) 'Tárcim
130 ' 128.129 és 130.131 kód definiálása
140 POKE C,0,0,2,2,62,94,30,20,36,35,0,0,0,0,0
145 POKE C+16,0,32,68,108,126,120,120,20,36,35
150 POKE 16459,C-256*INT(C/256),INT(C/256)
160 DIM LOx(5) 'Lovak pozíciója
170 FOR C=0 TO 5 : LOx(C)=3 : NEXT C : CLS
180 PRINT CHR$(2)CHR$(4)" L ö v e r s e n y "CHR$(18)CHR$(20)
190 PRINT $4,41,"C" : PRINT $7,41,"é" : PRINT $10,41,"L"
200 FOR C=35 TO 160 : SET(248,C) : NEXT C
210 FOR C=1 TO 5 : PRINT $2*C+1,0,C;CHR$(128)CHR$(130) : NEXT C
220 POKE 16443,PEEK(16443) AND 127 'NMI tilos
230 RANDOM : PRINT CHR$(6) 'Előtörles
240 FOR C=0 TO 500 : NEXT C
250 GOSUB 360 : GOSUB 360 : BEEP 50,400
260 'Lovak mozgatása
270 Vx=RND(5) : IF Vx=0 THEN 270
280 PRINT $2*Vx+1,LOx(Vx),CHR$(5)" "CHR$(21)CHR$(128)CHR$(130)
290 BEEP 50+10*Vx,3 'Mozgás hangja
300 LOx(Vx)=LOx(Vx)+1
310 IF LOx(Vx)<>39 THEN 270 'Tovább!
320 PRINT $14,8,"Győzött a"Vx". pályán futó ló!"
330 FOR C=0 TO 2000 : NEXT C : PRINT CHR$(22)
340 POKE 16443,PEEK(16443)+128 : OUT 0,PEEK(16443) 'NMI mehet!
350 GOTO 170 'Következő futam indítása
360 BEEP 50,50 : FOR C=0 TO 100 : NEXT C : RETURN
370 END 'Löverseny

```

----- FELJEGYZÉSEK -----

A PRIMO hanggenerátor paraméterezése

A PRIMO hanggenerátora elsősorban azt a célt szolgálja, hogy a billentyűk megérintését akusztikusan visszajelentse. Ezen túlmenően azonban egyszólamú dallamok megszólaltatására is felhasználható. Ehhez természetesen a hanggenerátort működtető BEEP BASIC utasítást úgy kell paraméterezni, hogy a megszólaló hang az egyenletesen temperált skála egyik eleme legyen. A következőkben megadjuk a paraméterek kiszámításához szükséges képleteket, és az egyenletesen temperált skála Kontra oktávjától a Kétvonalas oktávig terjedő - 1 másodperc hosszúságú - hangok keltéséhez szükséges BEEP-paraméterek értékét.

A PRIMO a BEEP utasítás hatására 50 % kitöltési tényezőjű négyszögjelet bocsájt ki. A megszólaló hang - mikroszekundumban mért - periódusidejét az alábbi képlet adja meg:

$$T=2*(8.4*X+35)$$

ahol X a BEEP utasítás első paramétere. A periódusidőből a hang frekvenciája - Hz-ben - a következő egyszerű számítással határozható meg:

$$f=1/T*10^6$$

A kiadott hang - másodpercben mért - hosszúságát a következőképpen számolhatjuk:

$$H=1/f*N$$

ahol N a BEEP utasítás második paramétere. Ebből a képletből adódik, hogy azonos hosszúságú, de különböző frekvenciájú hangokhoz más és más N érték tartozik.

Az egyenletesen temperált skála Kontra oktávjától a Kétvonalas oktávig terjedő zenei hangokhoz tartozó X és N értékeket adja meg az alábbi táblázat. Az első oszlopban a hang neve, a másodikban a hang névleges frekvenciája található. A harmadik oszlopban az az X érték áll, amellyel a BEEP utasítás a névleges frekvenciát legjobban megközelítő hangot kelti. A negyedik oszlopban található N értékkel 1 másodpercig szól az adott hang. Végül az utolsó oszlop azt mutatja, hogy az adott X érték esetén a PRIMO milyen frekvenciájú hangot ad ki. Ez általában nem egyezik meg a névleges frekvenciával, mivel X értéke csak egész szám lehet. Az eltérés a frekvencia növekedésével egyre nagyobb lesz.

----- FOGSÉLEK -----

	Hang	Névleges fr. [Hz]	X	N	PRIMO fr. [Hz]
Kontra oktáv					
	C	32.70	1816	32	32.70
	#C	34.65	1714	34	34.64
	D	36.71	1617	36	36.72
	#D	38.89	1526	38	38.90
	E	41.20	1440	41	41.22
41	F	43.65	1359	43	43.67
	#F	46.25	1283	46	46.24
	G	49.00	1211	48	48.98
	#G	51.91	1142	51	51.93
	A	55.00	1078	55	55.00
	#A	58.27	1017	58	58.29
	H	61.74	960	61	61.74
Nagy oktáv					
	C	65.41	906	65	65.40
	#C	69.30	855	69	69.28
	D	73.42	807	73	73.38
	#D	77.78	761	77	77.79
#	E	82.41	718	82	82.42
	F	87.31	678	87	87.26
	#F	92.50	639	92	92.55
	G	98.00	603	98	98.04
	#G	103.83	569	103	103.85
	A	110.00	537	109	109.99
	#A	116.54	507	116	116.45
	H	123.47	478	123	123.45
Kis oktáv					
	C c	130.81	451	130	130.77
	#C cis	138.59	425	138	138.70
	D d	146.83	401	146	146.91
	#D dis	155.56	378	155	155.75
	E e	164.81	357	164	164.81
a	F f	174.61	337	174	174.47
	#F fis	185.00	318	184	184.76
	G g	196.00	300	195	195.69
	#G gis	207.65	282	208	208.00
	A a	220.00	266	220	220.32
	#A ais	233.08	251	233	233.27
	H h	246.94	237	246	246.82

----- FÖGGELÉK -----

	Hang	Névleges fr. [Hz]	X	N	PRIMO fr. [Hz]
Egyvonalas oktáv	C	261.62	223	262	262.03
	#C	277.18	211	276	276.64
	D	293.66	199	292	292.98
	#D	311.13	187	311	311.37
	E	329.63	176	330	330.38
	F	349.23	166	349	349.80
	#F	369.99	157	369	369.33
	G	391.99	148	391	391.18
	#G	415.30	139	415	415.77
	A	440.00	131	440	440.37
	#A	466.16	124	464	464.43
	H	493.88	116	495	495.34
Kétvonalas oktáv	C	523.25	110	521	521.38
	#C	554.36	103	555	555.43
	D	587.33	97	588	588.37
	#D	622.25	91	625	625.47
	E	659.25	86	660	660.15
	F	698.45	81	698	698.91
	#F	739.98	76	742	742.50
	G	783.99	72	781	781.49
	#G	830.60	67	836	836.40
	A	880.00	63	886	886.21
	#A	932.32	60	927	927.64
	H	987.76	56	989	989.32

A fent megadott összefüggés X és a hang frekvenciája között csak akkor pontos, ha a BEEP utasítás végrehajtása alatt az NMI ki van kapcsolva. (Ha az NMI-t nem kapcsolnánk ki, akkor ezek az Interruptok megszakítva a frekvenciát meghatározó gépi ciklusokat, meghamisítják a periódusidő pontos értékét. Ez a tiszta hangban egy jól hallható kattogást eredményezne. A hang megszólaltatása után az NMI-t azért célszerű újra engedélyezni, hogy a PRIMO működését a RESET billentyű megnyomásával ismét megszakíthassuk.) Az NMI ki- és bekapcsolását mutatja be az alábbi BASIC programrészlet:

```

...
100 POKE 16443,PEEK(16443) AND 127
110 BEEP X,N
120 POKE 16443,PEEK(16443) OR 128 : OUT 0,PEEK(16443)
...

```

----- FELJEGYZÉSEK -----

A PRIMO memória felosztása

Az alábbiakban részletesen ismertetjük a PRIMO memória felosztását. Az ábra bal oldalán látható számértékek az adott terület kezdőcímét, a jobb oldalon álló számok a végcímét adják meg. Az egyes területek kezdő- és végcímét decimális és hexadecimális számrendszerben is feltüntettük. A számpár első tagja a decimális, második tagja a hexadecimális érték.

A számpárok némelyike zárójelben áll. Ez azt kívánja jelölni, hogy a kérdéses terület kezdőcíme változó érték, amely a PRIMO működése során - a végrehajtott BASIC program függvényében - más és más lehet. Ezért a zárójelben annak a BASIC Interpreter változónak a címe található, amely a kérdéses terület pillanatnyi kezdő-, vagy végcímét tárolja.

A számpárok közül hármat * karakterrel jelöltünk meg. E területek tipikus kezdőcíme az ábrából megállapítható. Ezek a címek a PRIMO BASIC Interpreterének működése során csak akkor változnak meg, ha erről a PRIMO Felhasználója gondoskodik. A három terület kezdőpontját a következő címeken elhelyezkedő BASIC rendszer- ill. Interpreter változók határozzák meg:

- BASIC program kezdőcíme (16548;40A4)
- Magnetofon puffer kezdőcíme (16474;405A)
- Billentyűzet puffer kezdőcíme (16551;40A7)

Az ábrában + karakterrel jelöltük azon területek címeit, amelyek a PRIMO memóriájának méretétől függenek. Az alábbi táblázatból a különféle típusú PRIMO-kra jellemző értékek kiolvashatók. (Mint az ábrából látható, e területek kezdő-, ill. végcímét a 16561;40B1 címen elhelyezkedő BASIC Interpreter változó tárolja, így a Felhasználó azt megváltoztathatja. A táblázat az értékek alapértelmezését tartalmazza.)

	PRIMO A-32	PRIMO A-48	PRIMO A-64
Karakter terület végcíme:	26623;67FF	43007;A7FF	59391;E7FF
Képernyő RAM kezdőcíme:	26624;6800	43008;A800	59392;E800
Képernyő RAM végcíme:	32767;7FFF	49151;BFFF	65535;FFFF

(A táblázatban megadott RAM-címek a PRIMO '84.1 verziójára vonatkoznak. Az értékek a PRIMO továbbfejlesztésekor megváltozhatnak.)

----- FÖSSZELÉK -----

		Képernyő RAM	← (16561;40B1)+6143 ⁺
(16561;40B1) ⁺	→	Karakterváltzó terület	← (16561;40B1)-1 ⁺
(16544;40A0)	→	STACK (veremtár)	← (16544;40A0)-1
(16616;40EB)	→	↓ ↓ ↓ ↓ Szabad terület	← (16616;40EB)-1
(16637;40FD)	→	↑ ↑ ↑ ↑	← (16637;40FD)-1
		Tömb változók	
(16635;40FB)	→	Skalár változók	← (16635;40FB)-1
(16633;40F9)	→	BASIC program	← (16633;40F9)-1
17385;43E9 [*]	→	Magnetofon puffer	← 17384;43EB
17129;42E9 [*]	→	Billentyűzet puffer	← 17128;42EB
16872;41EB [*]	→	BASIC Interpreter változók	← 16871;41E7
16512;4080	→	PRIMO rendszerváltozók	← 16511;407F
16384;4000	→	PRIMO BASIC ROM	← 16383;3FFF
00000;0000	→		

PRIMO rendszerváltozók

A **PRIMO** rendszerváltozói a 16384..16511 (4000...407F) címen helyezkednek el. Itt találhatók az **RST**-vektorok, néhány **ROM**-kapocs (**ROM-hook**), valamint a periféria kezelő programok munkaváltozói. A 128 byte méretű területen még fel nem használt szabad területek is találhatók. E területek későbbi fejlesztés céljára fenntartott részek, ezért alkalmazásukat nem ajánljuk.

A következőkben felsoroljuk a **PRIMO** rendszerváltozókat. Megadjuk decimális és hexadecimális memóriacímeiket, a változó által lefoglalt memória méretét, valamint a változó jelentését. Néhány esetben a változó után egy **B**, ill. **W** betű látható. Ez azt jelzi, hogy a kérdéses változó 8 bites byte-ként, ill. 16 bites szóként kerül felhasználásra. Ha a változó bitjei külön jelentéssel bírnak, a leírásban az egyes bitek értelmezése is megtalálható. (Egy byte legértékesebb helyiértékű bitje a 7-es bit.)

DDDDD	A változó decimális címe
HHHH	A változó hexadecimális címe
LL	A változó hossza
M	A változó felhasználási módja

DDDDD HHHH LL M értelmezés

16384	4000	3	RST 08H vektor	Az RST -vektorok a memória 8H, 10H,...38H címein lévő JP utasítások ROM -kapcsai. Segítségükkel a felhasználói programok is alkalmazhatnak RST -ket. (A PRIMO BASIC használja az RST 08H...20H utasításokat.)
16387	4003	3	RST 10H vektor	
16390	4006	3	RST 18H vektor	
16393	4009	3	RST 20H vektor	
16396	400C	3	RST 28H vektor	
16399	400F	3	RST 30H vektor	
16402	4012	3	RST 38H vektor	
16405	4015	3	ROM -kapocs a memória 5H címéről. (5H = JP 4015H)	
16408	4018	3	NMI -rutin vége	
			Az NMI -rutin működése befejeztével erre a címre ugrik, így a Felhasználó saját programjával bővítheti a Nem Maszkolható Interrupt rutint.	
16411	401B	3	Fenntartva	
16414	401E	3	RESET -rutin vége	
			A RESET -rutin működése után erre a címre ugrik, így a Felhasználó saját programjával bővítheti a RESET gomb megnyomása után lefutó programot.	
16417	4021	6	Fenntartva	

----- FOGGELÉK -----

16420 4024 3

16423 4027 3 Billentyűzet különleges mód ROM-kapocs
A GLINE rutin működése során, egy 128...255 kódú karakter begépelésekor a vezérlés erre a címre kerül. Így a Felhasználó saját programjával dolgoztathatja fel a különleges kódú karaktereket.

16426 402A 15 Fenntartva

16441 4039 2 W Képernyő RAM kezdőcíme

16443 403B 1 B KI-1 kapu tükör-byte
A PRIMO ebben a byte-ban tárolja a 0...63 című kapura utoljára kiírt értéket. F39.σ

16444 403C 1 Fenntartva

16445 403D 3 Real-time óra

A PRIMO működése során - ha az NMI aktiv - e számláló értékét minden 1/50-ed másodpercben 1-el megnöveli. A számláló legkisebb helyiértékű 8 bitje a 16445-ös címen helyezkedik el.

16448 4040 2 Fenntartva

A Z-80 mikroprocesszor IX regisztere a PRIMO működése közben mindvégig a 16450 (4042H) értéket tartalmazza. Az IX regiszter tartalmát a Felhasználó gépi programja még időszakosan sem változtathatja meg.

16450 4042 1 B Utolsóként megnyomott billentyű ASCII kódja

16451 4043 1 B Billentyűzet auto-repeat számláló

16452 4044 1 B Upper-bit, utolsó megnyomott billentyű címe

16453 4045 1 B Állapotjelző byte

Bit3 =1, Printer 1EH, 1FH konverzió tilos

Bit2 =1, Világít a kurzor

Bit0 =1, Billentyűzet különleges üzemmód aktiv ↓ 'él

Bit4 = BRK tiltva = 1!?

16454 4046 1 B Billentyűzet időzítés számláló (=24)

16455 4047 1 B Képernyőkezelő állapotjelző byte

Bit7 =1, Nyújtott méretű karakter megjelenítés

Bit6 =1, Karakter aláhúzás

Bit5 =1, Fehér képernyő alapszín

Bit4 =1, Inverz karakteralap

Bit3 =1, Karakter előtörölés

? → Bit2 =1, Karakter előtörölés puffer

Bit1 =1, Inverz rajzolás tirtaszine

Bit0 =1, Függőleges írás

16456 4048 1 B Képernyőkezelő karakter oszlop cím (0...255)

16457 4049 1 B Képernyőkezelő karakter sor cím (0...191)

16458 404A 1 B Képernyő RAM kezdőcímeinek értékesebb 8 bitje

16459 404B 2 W Kiegészítő karaktergenerátor ROM cím

A PRIMO a 128...255 kódú karakterek megjelenítés-

- sekor a karakter pontmintázatát tároló táblázat kezdőcímét ebből a változóból olvassa ki. Így e karaktereket a Felhasználó definiálhatja.
- 16461 404D 3 Fenntartva
- 16464 4050 1 B A nyomtató egy sorának maximális hossza
- 16465 4051 1 B A nyomtató egy lapján lévő sorok maximális száma
Ha egy lapon több mint 127 sort jelölünk ki, a PRIMO a nyomtatóra nem küld lapdobás (Form Feed) karaktert.
- 16466 4052 1 B A nyomtató aktuális sorában lévő karakterek száma
- 16467 4053 1 B A nyomtató aktuális lapján lévő sorok száma
- 16468 4054 1 B A kurzor nyomtató soron belüli pozíciója
- 16469 4055 1 Fenntartva
- 16470 4056 1 B Magnó állapotjelző byte
Bit7 =1, Van megnyitott magnó állomány Bit3 =1 nincs hang felvételkor
Bit6 =1, Az állomány olvasásra nyitott
Bit5 =1, Magnó puffer ~~üres~~
Bit4 =1, TEST utasítás végrehajtása folyik
=0, LOAD utasítás végrehajtása folyik
Bit2 =1, Megadtak állománynevet
Bit1 =1, Információs sor nem jelenik meg
- 16471 4057 1 B Magnó puffer számláló (0=256)
- 16472 4058 2 W Magnó puffer pointer
- 16474 405A 2 W Magnó puffer kezdőcíme
- 16476 405C 1 B Feldolgozás alatt álló blokk típusa
083H = BASIC program név-blokk 131
087H = Adatállomány név-blokk 135
177 0B1H = BASIC program záró-blokk
181 0B5H = Képernyő tartalom záró-blokk
183 0B7H = Adatállomány záró-blokk
185 0B9H = Gépi kódú program záró-blokk
241 0F1H = BASIC program adat-blokk
245 0F5H = Képernyő tartalom adat-blokk
247 0F7H = Adatállomány adat-blokk
249 0F9H = Gépi kódú program adat-blokk
EZ NINCS benne
- 16477 405D 1 B Hibásan beolvasott blokkok száma (BCD kódban)
- 16478 405E 1 B Utoljára beolvasott blokk sorszáma (BCD kódban)
- 16479 405F 1 B Magnó állapotjelző byte
010H = Az adott magnó nem fordít polaritást ←BRG
0EFH = Az adott magnó polaritást fordít
- 16480 4060 1 B Magnó beolvasáskor számított bit-ideje (~75) 315 MHz 50
- 16481 4061 1 Fenntartva
- 16482 4062 16 B Magnó állománynév puffer
- 16498 4072 14 Fenntartva

16511 407F PRIMO rendszerváltozó terület végcíme

(A táblázatban megadott RAM-címek a PRIMO '84.1 verziójára vonatkoznak. Az értékek a PRIMO továbbfejlesztésekor megváltozhatnak.)

----- FELJEGYZÉSEK -----

PRIMO BASIC Interpreter változók

A PRIMO Interpreter változói a 16512...16868 (4080...41E4) címeken helyezkednek el. Itt találhatók a BASIC interpreter működéséhez szükséges munkaváltozók, az aritmetika munkaváltozói, az Interpreter ROM-kapcsai, valamint a '84.1 verzió által nem értelmezett BASIC kulcsszavak ROM-kapcsai. A BASIC Interpreter változók között még fel nem használt, szabad területek is találhatók. Ezek a későbbi fejlesztések számára fenntartott területek, ezért alkalmazásukat nem ajánljuk.

A következőkben felsoroljuk a PRIMO Interpreter változóit. Megadjuk decimális és hexadecimális memóriacímeiket, a változók által lefoglalt memória méretét, valamint a változók jelentését. A változók után egy B, ill. W betű is látható. Ez azt jelzi, hogy a kérdéses változót a PRIMO 8 bites byte-ként, vagy 16 bites szóként értelmezi-e.

DDDDD	A változó decimális címe
HHHH	A változó hexadecimális címe
LL	A változó hossza
M	A változó felhasználási módja

DDDDD HHHH LL M értelmezés

16512	4080	14	<u>Kivonórutin</u> az egyszerespontos osztás számára. Az egyszerespontos osztó rutin működésének egyszerűsítése érdekében ezen a területen – az aktuális osztó függvényében változó – szubrutin helyezkedik el. Az utasítás kódokat részben a PRIMO bekapcsolásakor működő inicializáló program, részben maga az osztó rutin írja be erre a területre.
16526	408E	2	Fenntartva
16528	4090	3	<u>Véletlenszám generátor</u> szorzó. Ezen a területen a PRIMO az 5105216 értékű konstanst tárolja. Egy újabb véletlen szám generálásakor a PRIMO ezt az értéket megszorozza az előző véletlen számmal, majd a szorzat kevésbé értékes 24 bitjéből alakítja ki a soronkövetkező véletlen számot.
16531	4093	3	Szubrutin az INP függvény számára. A rutin az IN A,(x) -- RET utasításokból áll. Az INP függvény argumentumaként megadott címet – x –

----- FÜGGELÉK -----

			az IN utasításba az INP függvényt végrehajtó programrész helyezi el.
16534	4096	4	Fenntartva
16538	409A	1 B	Utolsó hiba kódja
16539	409B	1	Fenntartva
16540	409C	1 B	Aktuális output eszköz kódja képernyő = 0 printer = 1 $\rightarrow \emptyset$ magnó = 1 $\rightarrow \emptyset$
16541	409D	1 B	Képernyő sor hossza (=42)
16542	409E	2	Fenntartva
16544	40A0	2 W	Karakterváltozó terület végcime
16546	40A2	2 W	Aktuális BASIC programsor sorszáma
16548	40A4	2 W	BASIC program terület kezdőcime
16550	40A6	1	Fenntartva
16551	40A7	2 W	Billentőzet puffer kezdőcime (=16872;41E8)
16553	40A9	1 B	INPUT utasítás állapotjelző byte <i>Szoftverhiba #002 nem olvas.</i> = 0, INPUT a billentyűzetről = -1, INPUT a magnóról <i>Poke 16553, 1 megoldja.</i>
16554	40AA	3	Utolsó generált véletlen szám
16557	40AD	1	Fenntartva
16558	40AE	1 B	Változó keresés/definíálás állapotjelző = 0, csak keresés, definíálás tilos $\neq \emptyset$, sikertelen keresés után definíálás
16559	40AF	1 B	Lebegőpontos akkumulátorban őrzött érték típusa = 2, egész típusú érték = 3, karakteres érték = 4, egyszerespontos érték = 8, duplapontos érték
16560	40B0	1 B	Többcélú puffer változó
16561	40B1	2 W	BASIC által használható RAM végcime+1
16563	40B3	2 W	Karakter-operandus STACK mutató
16565	40B5	30	Karakter-operandus STACK
16595	40D3	3	Utolsó karakter-operandus leírója
16598	40D6	2 W	Karakterváltozó terület mutató
16600	40D8	2	Többcélú puffer változó
16602	40DA	2 W	Aktuális DATA sor sorszáma
16604	40DC	1 B	FOR utasítás állapotjelző = 0, nem FOR utasítás végrehajtása folyik
16605	40DD	1 B	Konstans 0 <i>nem konstans! Return vez.</i>
16606	40DE	1 B	INPUT/READ állapotjelző = 0, INPUT utasítás végrehajtása folyik
16607	40DF	2	Többcélú puffer változó
16609	40E1	1 B	AUTO állapotjelző = 0, AUTO üzemmód nem aktiv
16610	40E2	2 W	AUTO kezdő/soronkövetkező sorszám
16612	40E4	2 W	AUTO növekmény (kezdetben = 10)
16614	40E6	2 W	Aktuális BASIC utasítás programterületen belüli kezdőcime
16616	40E8	2 W	Aktuális BASIC utasítás végrehajtásának megkezdé-

			sekori STACK mutató értéke
16618	40EA	2 W	Utolsó <u>hibát</u> tartalmazó BASIC programsor <u>sorszáma</u>
16620	40EC	2 W	Az utolsó hibát tartalmazó, ill. az utoljára listázott/editált sor sorszáma (a . értéke)
16622	40EE	2 W	Utolsó <u>hibát</u> okozó BASIC utasítás programterületen belüli kezdőcíme
16624	40F0	2 W	Programozott hibakezelést végző BASIC rutin programterületen belüli kezdőcíme
16626	40F2	1 B	Programozott hibakezelés állapotjelző = 0, nem programozott hibakezelés végrehajtása folyik <i>255 Programozott hibakezelés folyik</i>
16627	40F3	2 W	Programterület mutató puffer A PRIMO egy kifejezés kiértékelése idején itt tárolja a programterület mutató aktuális értékét.
16629	40F5	2 W	CONT sorszám A BASIC program megszakítása, ill. STOP/END utasítással történt megállítása után a PRIMO itt tárolja azon BASIC utasítássor sorszámát, amelyben az utoljára végrehajtott BASIC utasítás elhelyezkedik.
16631	40F7	2 W	CONT programterület címe A BASIC program megszakítása, ill. STOP/END utasítással történt megállítása után a PRIMO itt tárolja a soronkövetkező BASIC utasítás programterületen belüli kezdőcímét.
16633	40F9	2 W	Skalár változók táblázatának kezdőcíme
16635	40FB	2 W	Tömb változók táblázatának kezdőcíme
16637	40FD	2 W	Szabad BASIC programterület kezdőcíme
16639	40FF	2 W	DATA -lista mutató
16641	4101	26 B	Változó típusazonosító tábla
16667	411B	1 B	TRON/TROFF állapotjelző = 1, TRON aktív <i>≠ 0 aktív</i>
16668	411C	1 B	Atmeneti tároló az aritmetika számára
16669	411D	8	Lebegőpontos Akkumulátor #1
16677	4125	1 B	Atmeneti tároló az eredmény előjele számára
16678	4126	1 B	Atmeneti tároló az aritmetika számára
16679	4127	8	Lebegőpontos Akkumulátor #2
16687	412F	1	Fenntartva
16688	4130	26	Munkaterület a Bináris → Karakteres konverzió számára
16714	414A	8	Atmeneti tároló a duplapontos osztó rutin számára <i>Valós * is használja</i>

----- FOGGELÉK -----

A soronkövetkező területen azon BASIC utasítások ROM-kapcsai helyezkednek el, melyek a PRIMO '84.1 verziójában nem definiáltak.

A 16722...16805 (4152...41A5) címeken JP utasítások helyezkednek el, melyek a vezérlést alaphelyzetben az SN - Syntax Error, Formai hiba - hibajelzést végrehajtó rutinra adják át. A Felhasználó a JP utasítások címmezőjét átírva, az egyes ROM-kapcsoknál megadott BASIC kulcsszavak előfordulásakor a vezérlést a saját gépi kódú rutinjára adhatja át, ily módon a PRIMO BASIC programozási nyelvét új utasításokkal, ill. függvényekkel bővítheti.

16722 4152	CVI	16764 417C	FIELD
16725 4155	FN	16767 417F	GET
16728 4158	CVS	16770 4182	PUT
16731 415B	DEF	16773 4185	Fenntartva
16734 415E	CVD	16776 4188	Fenntartva
16737 4161	EOF	16779 418B	MERGE
16740 4164	LOC	16782 418E	Fenntartva
16743 4167	LOF	16785 4191	KILL
16746 416A	MKIS	16788 4194	&
16749 416D	MKSS	16791 4197	Fenntartva
16752 4170	MKDS	16794 419A	Fenntartva
16755 4173	CMD	16797 419D	INSTR
16758 4176	TIMES	16800 41A0	Fenntartva
16761 4179	Fenntartva	16803 41A3	Fenntartva

A következő területen olyan ROM-kapcsok helyezkednek el, melyek a ROM különböző pontjairól a vezérlést a RAM-ba adják át, lehetővé téve, hogy a PRIMO BASIC programozási nyelv egyes utasításainak működését a Felhasználó saját gépi kódú programrész beiktatásával megváltoztassa.

Az eddigiekhez hasonlóan megadjuk a ROM-kapocs decimális és hexadecimális címét, valamint azt a hexadecimális ROM-címet, ahonnan a vezérlés - egy CALL gépi utasítás segítségével - az adott ROM-kapocsra kerül. Röviden arra is utalunk, hogy a kiugrás milyen funkció végrehajtásának megkezdése előtt következett be. (A 16806...16868 (41A6...41E4) címeken RET utasítások helyezkednek el. A Felhasználó feladata, hogy a kiválasztott címre egy JP utasítást írjon, amely a vezérlést a saját gépi kódú rutin elejére adja át.)

16806	41A6	- 19EC	- Hibajelzés, (E)=hibakód
16809	41A9		- Fenntartva
16812	41AC	- 1A1C	- Visszatérés Direkt módba HIBQ, BRK, END
16815	41AF		- Fenntartva
16818	41B2	- 1AA1	- Tokenizálás vége, végrehajtás kezdete
16821	41B5	- 1AEC	- Új sor beinzertálása befelyeződött
16824	41B8	- 1AF2	- Új sor beinzertálását követő NEW parancs végrehajtása befejeződött
16827	41BB	- 1B8C	- NEW parancs végrehajtásának vége
16830	41BE	- 2174	- PRINT, LPRINT utasítás vége
16833	41C1	- 032C	- Kiírás az aktuális output eszközre
16836	41C4		- Fenntartva
16839	41C7	- 1EA6	- Operandussal kiegészített RUN parancs kezdete
16842	41CA	- 206F	- PRINT utasítás kezdete
16845	41CD	- 20C6	- PRINT, LPRINT utasításban egy szám kiírása az aktuális output eszközre
16848	41D0	- 2103	- Soremelés az aktuális output eszközön
16851	41D3	- 2108	- PRINT, LPRINT utasításban vessző feldolgozás kezdete
16854	41D6	- 219E	- INPUT utasítás kezdete
16857	41D9		- MIDS utasítás
16860	41DC	- 222D	- INPUT, READ utasításban a következő változó értékadásának előkészítése
16863	41DF	- 227B	- nem magnóról végrehajtott INPUT utasítás vége
16866	41E2		- Fenntartva

(A táblázatokban megadott RAM- és ROM-címek a PRIMO '84.1 verziójára vonatkoznak. Az értékek a PRIMO továbbfejlesztésekor megváltozhatnak.)

----- FELJEGYZÉSEK -----

A PRIMO Bemeneti/Kimeneti kapui (I/O Portok)

A PRIMO a 256 Bemeneti és Kimeneti kaput 4 - egyenként 64 címet tartalmazó - csoportra osztja, és az így keletkezett 4 Bemenet és 4 Kimenet közül az első két B/K kaput használja. A következőkben felsoroljuk e kapuk bitjeinek egyedi jelentését. (Az egyes bitek által vezérelt áramkörök részletes ismertetése a PRIMO FÖZETEK sorozatban megjelent HARDVER leírásban található.)

Kimeneti kapuk

A PRIMO két kimeneti kaput - KI-1, KI-2 - használ. A KI-1 8 bites, a KI-2 4 bites. A PRIMO '84.1 verziója csak a KI-1 kapu bitjeit kezeli, a másik kapu kezelését - ha szükséges - gépi kódú, vagy BASIC programmal a Felhasználónak kell megoldania.

KI-1 (0...63 címek)

default kiírás 237

A KI-1 kapu a PRIMO általános kimeneti regisztere. A kapu 8 bitje segítségével a hardver különféle részeit a többbitől függetlenül vezérelhetjük. Az egyes bitek jelentése a következő:

Bit7: NMI engedélyezés

Ha e bit értéke =1, akkor a PRIMO működését minden ~~1/50~~ 1/100 másodpercben egy NMI szakítja meg. Ellenkező esetben az NMI nem jut érvényre.

Bit6: Botkormány léptetés

Ezen bit értékének minden 1→0 átmenete a botkormányba épített kiválasztó-számláló tartalmát 1-el megnöveli.

Bit5: Magnó távvezérlés, vagy V24/2

A bitbe 0 értéket írva a PRIMO hátlapján lévő "TAPE" jelű csatlakozó 4-es pontján +5V feszültség jelenik meg, míg a bit 0 értéke esetén a csatlakozón a Földpotenciálhoz viszonyítva 0V mérhető.

Bit4: Hanggenerátor - Duda - vezérlés

A bitbe 1 értéket írva a PRIMO a piezoelektromos hangszóró membránját "elmozdítja", míg 0 érték esetén nyugalomban hagyja.

----- FÖGGELEK -----

Bit3: Képernyő lapkijelölés

Ha a bit értéke 1, a PRIMO a számítógéphez kapcsolt TV készüléken a memória utolsó 6 Kbyte-ja által meghatározott képtartalmat jeleníti meg, míg ha a bit 0 értékű, a TV-n a memória 8 Kbyte-al alacsonyabb címétől kezdődő 6 Kbyte-nyi területe jelenik meg.

Bit2: Magnó távvezérlés, vagy V24/1 *Led*

A bitbe 0 értéket írva a PRIMO hátlapján lévő "TAPE" jelű csatlakozó 5-ös pontján +5V feszültség jelenik meg, míg a bit 1 értéke esetén a csatlakozón a Földpotenciálhoz viszonyítva 0V mérhető.

Bit1: Magnó adatkimenet vezérlés 1

Bit0: Magnó adatkimenet vezérlés 2

Az 1-es és 0-as bitek a magnó adatkimenetének vezérlését szolgálják. A két bit együttes értéke határozza meg a PRIMO "TAPE" jelű csatlakozója 1-es kimeneti pontjának állapotát. A különböző állapotok a következő kódokkal válthatók ki:

Bit1: 0	Bit0: 0	alacsony szint	~ -110mV
0	1	alaphelyzet	~ 0V
1	0	alaphelyzet	~ 0V
1	1	magas szint	~ +110mV

A KI-1 kapu egyes bitjeinek független jelentése van, de a 8 bit csak egyszerre írható, és a kapu aktuális tartalma nem olvasható vissza. Ezért a memóriában a 16443;403B címen a rendszer eltárolja a KI-1 kapura utoljára kiírt 8 bit értékét. Ezt az értéket tükör-byte-nak nevezzük. Amennyiben a KI-1 kapu valamelyik bitjét módosítani akarjuk, azt helyesen úgy tehetjük meg, hogy a tükör-byte értékét a memóriából kiolvassuk, abban a kiválasztott bit értékét megváltoztatjuk, majd az új értéket egyrészt visszairjuk a memóriába, másrészt kiküldjük a 0...63 című hardver kapura. Ha nem így járunk el, a PRIMO helytelenül fog működni. Az előbbi folyamat BASIC programmal történő megvalósítására tekintünk a következő példákat:

Feladat: Az NMI kikapcsolása

```
POKE 16443,PEEK(16443)AND 127 'Bit7-be 0
OUT 0,PEEK(16443) 'Tükör-byte kiküldése
```

Feladat: Az NMI visszakapcsolása

```
POKE 16443,PEEK(16443) OR 128 : OUT 0,PEEK(16443)
```

KI-2 (64...127 CIMEK)⁺

A KI-2 kapu a PRIMO opcionális soros buszát vezérli. E soros busz felhasználásával pl. a PRIMO-hoz - megfelelő program segítségével - a Commodore személyi számítógépeknél alkalmazott soros buszra illeszkedő perifériákat kapcsolhatunk. A soros busz kezelését a '84.1 verzió nem teszi lehetővé, azt gépi kódú, vagy BASIC programmal a Felhasználónak kell megoldania. A soros busz egyes bitjeinek jelentése a következő:

Bit5: Soros busz órajel (SCLK) 32

Az SDATA vonalat érvényesítő jel. Mivel a soros buszon az információk byte-okként értelmezettek, e jel mint a párhuzamos-soros átalakító órajele alkalmazható.

Bit4: Soros busz adat (SDATA) 16

A soros busz adatjele. Ezen a vonalon keresztül küldhető ki - párhuzamos-soros átalakítás után - adat a megcímzett perifériára.

Bit2: Fenntarva (SRQ) 4

Soros busz kiszolgálás kérés. A megfelelő funkció felhasználásához a PRIMO nyomtatott áramköri lemezén egy átkötést - jumper - kell létrehozni. (Az áramkör részletes ismertetése a HARDVER leírásban található meg. Az SRQ jel segítségével a PRIMO-ban maszkolható IT generálható.)

Bit1: Soros busz figyelemfelhívás (ATN) 2

E jel aktivizálásával a soros buszra kapcsolt valamennyi periféria arra utasítható, hogy - egy megfelelő protokoll szabályai szerint - vizsgálja: a buszon érkező információ számára értékes-e. Az a berendezés amelyik az ATN jelet aktiválja, lesz a BUS-MASTER, a többi periféria SLAVE üzemmódban fog működni.

Bit7:

Bit6:

Bit3:

Bit0: Ezen bitekhez nem kapcsolódik hardver.

A soros buszon az aktiv jelszint alacsony, vagyis az egyes bitek 0 értéke képviseli az aktiv, az 1 érték az inaktív állapotot.

+ A KI-2 KAPU CIMEI A PRIMO TOVÁBBFEJLESZTÉSÉVEL A 128...191 INTERVALLUMBA FOGNAK KERÜLNI.

Bemeneti kapuk

A PRIMO két bemeneti kaput - BE-1, BE-2 - használ. Mindkét bemeneti kapu 6 bites. A PRIMO '84.1 verziója csak a BE-1 kapu bitjeit kezeli, a másik kapu kezelését - ha szükséges - gépi kódú, vagy BASIC programmal a Felhasználónak kell megoldania.

BE-1 (0...63 címek)

A BE-1 kapu a PRIMO általános bemeneti regisztere. A kapu 6 bitje segítségével a hardver különféle részeinek állapota külön-külön lekérdezhető. Az egyes bitek jelentése a következő:

Bit5: Képkicoltás

32

A bit 1 értéke azt jelzi, hogy a PRIMO-hoz kapcsolt TV készülék elektronsugara - képvisszafutás miatt - ki van kapcsolva. E bit vizsgálatával a futó program működését a másodpercenként 50 alkalommal bekövetkező képváltáshoz szinkronizálhatja.

Bit4: BE-2/4 bit

16

A 4-es bit értékét a PRIMO "BIE" jelű - 2*25 pólusú - csatlakozójához kapcsolt berendezés határozhatja meg. A bit értéke a BIE B-15 pontjának állapotát tükrözi.

Bit3: BE-2/3 bit

8

A 3-as bit értékét a PRIMO "BIE" jelű - 2*25 pólusú - csatlakozójához kapcsolt berendezés határozhatja meg. A bit értéke a BIE B-25 pontjának állapotát tükrözi.

Bit2: Magnó adatbemenet

4

A magnó adatbemenet bit értékét a PRIMO "TAPE" jelű csatlakozójának 3-as pontján lévő feszültség jelformálás utáni állapota határozza meg. Ezen a biten keresztül fogadja a PRIMO a magnóról érkező soros adatbiteket. A bit alacsony értéke a PRIMO-ban maszkolható IT-t generál.

Bit1: RESET nyomógomb állapota

2

Az 1-es bit állapota a PRIMO hátoldalán elhelyezkedő RESET billentyű - pergésmentesítés utáni - aktuális értékét jelzi. A bit értékét a PRIMO NMI-kiszolgáló rutinja vizsgálja, de a Felhasználó gépi kódú, vagy BASIC program segítségével is ellenőrizheti.

nyomott gomb 1-et ad.

Bit0: Billentyűzet bemenet

A billentyűzet megcímzett gombjának aktuális állapotát jelképezi. Ha a bit értéke =1, a kiválasztott gombot éppen megérintettük.

A BE-1 kapu 5...1 bitjeivel ellentétben - ahol a beolvasáskor alkalmazott 0...63 címek egyenértékűek -, a különböző címekről olvasott 0-ás bit-értékek a billentyűzet egyes gombjainak állapotára jellemzőek. Így a Z-80 mikroprocesszor 0...63 című bemeneti kapuinak 0-ás bitjeit végigvizsgálva a billentyűzet valamennyi gombjának állapota lekérdezhető. (A billentyűzet gombjaihoz rendelt periféria címek a D. Függelékben találhatók.)

Bit7:

Bit6: Ezen bitekhez nem kapcsolódik hardver.

BE-2 (64...127 címek)

A BE-2 kapu a PRIMO különleges bemeneti regisztere. Ezen a kapun keresztül olvashatók be az - opcionális - soros busz, és a botkormányok állapota. Az egyes bitek jelentése a következő:

Bit5: Soros busz órajel (SCLK)

Az SDATA vonalat érvényesítő jel. Mivel a soros buszon az információk byte-okként értelmezettek, e jel mint a soros-párhuzamos átalakító órajele alkalmazható.

Bit4: Soros busz adat (SDATA)

A soros busz adatjele. Ezen a vonalon keresztül olvasható be - soros-párhuzamos átalakítás után - adat a megcímzett perifériáról.

Bit3: Soros busz kiszolgálás kérés (SRQ)

Az SRQ jel segítségével jelezheti egy - a soros buszra kapcsolódó - periféria, hogy a BUS-MASTER-től kiszolgálást kér. Az SRQ jel aktivizálása a PRIMO-ban maszkolható IT-t okoz.

Bit2: Botkormány adatbemenet 2

Ezen a bemeneti biten vizsgálható a PRIMO-hoz kapcsolt 2. botkormány 8 kapcsolójának pillanatnyi állapota. A megfelelő kapcsolót a Botkormány léptetés (KI-1/6 bit) kimeneten kiküldött impulzusok hatására értéket váltó kiválasztó-számláló aktuális értéke jelöli ki.

----- FOGBELÉK -----

Bit1: Soros busz figyelemfelhívás (ATN)

E jel aktivizálásával a PRIMO arra utasítható, hogy - egy megfelelő protokoll szabályai szerint - vizsgálja: a buszon érkező információ számára értékes-e. Ha a PRIMO az üzenetet értékesnek találja, a továbbiakban SLAVE üzemmódban várja a soros buszon érkező adatokat.

Bit0: Botkormány adatbemenet 1

Ezen a bemeneti biten vizsgálható a PRIMO-hoz kapcsolt 1. botkormány 8 kapcsolójának pillanatnyi állapota. A megfelelő kapcsolót a Botkormány léptetés (KI-1/6 bit) kimeneten kiküldött impulzusok hatására értéket váltó kiválasztó-számláló aktuális értéke jelöli ki.

Bit7:

Bit6: Ezen bitekhez nem kapcsolódik hardver.

E leírásban fel nem sorolt kapukat a PRIMO működtető programja nem használja, így azok a számítógéphez kapcsolt különböző perifériák vezérlésére használhatók. Az egyes kapu-csoportok hardver kiválasztását a PRIMO elvégzi, a dekódolt jelek a "BIE" jelű csatlakozón elérhetők. (A részletes ismertetés a HARDVER leírásban megtalálható.)

A PRIMO továbbfejlesztése során a 192...255 című be-, ill. kimeneti kapuk felhasználását nem tervezzük. Ezért ezek a címek szabadon felhasználhatók.

PRIMO BASIC kulcsszavak belső kódjai

A PRIMO BASIC Interpretere a Felhasználó BASIC programjának gyorsabb végrehajtása és memóriaigényének csökkentése érdekében a BASIC nyelv kulcsszavait és aritmetikai, ill. relációs műveleti jeleit a memóriában egyetlen 8 bites adat - úgynevezett token - formájában tárolja. A tokenek kódja a 128...251 tartományba esik. A tokenek pontos ismerete nélkül a BASIC program memórián belüli formája nehezen ismerhető fel, és a Felhasználó nem vállalkozhat a PRIMO BASIC nyelvének saját gépi programjával végrehajtott bővítésére.

Ezért a következőkben - két táblázatba rendezve - felsoroljuk a PRIMO BASIC valamennyi tokenjét, megadva annak decimális és hexadecimális kódját. A tokenek egy része a BASIC '84.1 verziójában nem alkalmazható. Ezeket a kulcsszavakat az alábbi táblázatban a * karakterrel jelöltük meg.

Kulcsszavak és műveleti jelek token kód szerint rendezve

token	kód	kulcsszó	token	kód	kulcsszó	token	kód	kulcsszó
dec	hex		dec	hex		dec	hex	
128	80	END	152	98	DEFSTR	176	B0	DEF *
129	81	FOR	153	99	DEFINT	177	B1	POKE
130	82	RESET	154	9A	DEFSNG	178	B2	PRINT
131	83	SET	155	9B	DEFDBL	179	B3	CONT
132	84	CLS	156	9C	BEEP	180	B4	LIST
133	85	CMD *	157	9D	EDIT	181	B5	LLIST
134	86	RANDOM	158	9E	ERROR	182	B6	DELETE
135	87	NEXT	159	9F	RESUME	183	B7	AUTO
136	88	DATA	160	A0	OUT	184	B8	CLEAR
137	89	INPUT	161	A1	ON	185	B9	CLOAD *
138	8A	DIM	162	A2	OPEN	186	BA	CSAVE *
139	8B	READ	163	A3	FIELD *	187	BB	NEW
140	8C	LET	164	A4	GET *	188	BC	TAB(
141	8D	GOTO	165	A5	PUT *	189	BD	TO
142	8E	RUN	166	A6	CLOSE	190	BE	FN *
143	8F	IF	167	A7	LOAD	191	BF	USING
144	90	RESTORE	168	A8	MERGE *	192	C0	VARPTR
145	91	GOSUB	169	A9	TEST	193	C1	CALL
146	92	RETURN	170	AA	KILL *	194	C2	ERL
147	93	REM	171	AB	CREATE	195	C3	ERR
148	94	STOP	172	AC	fn *	196	C4	STRINGS
149	95	ELSE	173	AD	SAVE	197	C5	INSTR *
150	96	TRON	174	AE	SCREEN	198	C6	POINT
151	97	TROFF	175	AF	LPRINT	199	C7	TIMES *

----- FÖGBELÉK -----

token dec	kód hex	kulcsszó	token dec	kód hex	kulcsszó	token dec	kód hex	kulcsszó
200	C8	PI	218	DA	FRE	235	EB	LOF *
201	C9	INKEY\$	219	DB	INP	236	EC	MKI\$ *
202	CA	THEN	220	DC	POS	237	ED	MKS\$ *
203	CB	NOT	221	DD	SQR	238	EE	MKD\$ *
204	CC	STEP	222	DE	RND	239	EF	CINT
205	CD	+	223	DF	LOG	240	F0	CSNG
206	CE	-	224	E0	EXP	241	F1	CDBL
207	CF	*	225	E1	COS	242	F2	FIX
208	D0	/	226	E2	SIN	243	F3	LEN
209	D1	↑	227	E3	TAN	244	F4	STR\$
210	D2	AND	228	E4	ATN	245	F5	VAL
211	D3	OR	229	E5	PEEK	246	F6	ASC
212	D4	)	230	E6	CVI *	247	F7	CHR\$
213	D5	=	231	E7	CVS *	248	F8	LEFT\$
214	D6	<	232	E8	CVD *	249	F9	RIGHT\$
215	D7	SGN	233	E9	EOF *	250	FA	MID\$
216	D8	INT	234	EA	LOC *	251	FB	'
217	D9	ABS						

A felsorolt szavak a BASIC kulcsszavai, ezért változónevek-
ként nem alkalmazhatók. A kulcsszavak a változónevek belsejében
sem fordulhatnak elő.

Kulcsszavak és műveleti jelek ABC sorrendbe rendezve

kulcsszó	token dec	kód hex	kulcsszó	token dec	kód hex	kulcsszó	token dec	kód hex
'	251	FB	END	128	80	OR	211	D3
*	207	CF	EOF	* 233	E9	OUT	160	A0
+	205	CD	ERL	194	C2	PEEK	229	E5
-	206	CE	ERR	195	C3	PI	200	C8
/	208	D0	ERROR	158	9E	POINT	198	C6
<	214	D6	EXP	224	E0	POKE	177	B1
=	213	D5	FIELD	* 163	A3	POS	220	DC
>	212	D4	FIX	242	F2	PRINT	178	B2
↑	209	D1	FOR	129	81	PUT	* 165	A5
ABS	217	D9	FN	* 190	BE	RANDOM	134	86
AND	210	D2	FRE	218	DA	READ	139	8B
ASC	246	F6	GET	* 164	A4	REM	147	93
ATN	228	E4	GOSUB	145	91	RESET	130	82
AUTO	183	B7	GOTO	141	8D	RESTORE	144	90
BEEP	156	9C	IF	143	8F	RESUME	159	9F
CALL	193	C1	INKEY\$	201	C9	RETURN	146	92
CDBL	241	F1	INP	219	DB	RIGHT\$	249	F9
CHR\$	247	F7	INPUT	137	89	RND	222	DE
CINT	239	EF	INSTR	* 197	C5	RUN	142	8E
CLEAR	184	B8	INT	216	D8	SAVE	173	AD
CLOAD	* 185	B9	KILL	* 170	AA	SCREEN	174	AE
CLOSE	166	A6	LEFT\$	248	F8	SET	131	83
CLS	132	84	LEN	243	F3	SGN	215	D7
CMD	* 133	85	LET	140	8C	SIN	226	E2
CONT	179	B3	LIST	180	B4	SQR	221	DD
COS	225	E1	LLIST	181	B5	STEP	204	CC
CREATE	171	AB	LOAD	167	A7	STOP	148	94
CSAVE	* 186	BA	LOC	* 234	EA	STR\$	244	F4
CSNG	240	F0	LOF	* 235	EB	STRING\$	196	C4
CVD	* 232	E8	LOG	223	DF	TAB(	188	BC
CVI	* 230	E6	LPRINT	175	AF	TAN	227	E3
CVS	* 231	E7	MERGE	* 168	A8	TEST	169	A9
DATA	136	88	MID\$	250	FA	THEN	202	CA
DEF	* 176	B0	MKD\$	* 238	EE	TIMES	* 199	C7
DEFDBL	155	9B	MKIS	* 236	EC	TO	189	BD
DEFINT	153	99	MKSS	* 237	ED	TROFF	151	97
DEFSNG	154	9A	NEW	187	BB	TRON	150	96
DEFSTR	152	98	NEXT	135	87	USING	191	BF
DELETE	182	B6	NOT	203	CB	VAL	245	F5
DIM	138	8A	ON	161	A1	VARPTR	192	C0
EDIT	157	9D	OPEN	162	A2	fn	* 172	AC
ELSE	149	95						

----- FELJEGYZÉSEK -----

Konverziós táblázatok

Hexadecimális - decimális konverziós tábla

16^3		16^2		16^1		16^0	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4 096	1	256	1	16	1	1
2	8 192	2	512	2	32	2	2
3	12 288	3	768	3	48	3	3
4	16 384	4	1 024	4	64	4	4
5	20 480	5	1 280	5	80	5	5
6	24 576	6	1 536	6	96	6	6
7	28 672	7	1 792	7	112	7	7
8	32 768	8	2 048	8	128	8	8
9	36 864	9	2 304	9	144	9	9
A	40 960	A	2 560	A	160	A	10
B	45 056	B	2 816	B	176	B	11
C	49 152	C	3 072	C	192	C	12
D	53 248	D	3 328	D	208	D	13
E	57 344	E	3 584	E	224	E	14
F	61 440	F	3 840	F	240	F	15

2 és 16 hatványai

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18

$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10

Decimális - hexadecimális konverziós tábla

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
10	A	B	C	D	E	F	10	11	12	13
20	14	15	16	17	18	19	1A	1B	1C	1D
30	1E	1F	20	21	22	23	24	25	26	27
40	28	29	2A	2B	2C	2D	2E	2F	30	31
50	32	33	34	35	36	37	38	39	3A	3B
60	3C	3D	3E	3F	40	41	42	43	44	45
70	46	47	48	49	4A	4B	4C	4D	4E	4F
80	50	51	52	53	54	55	56	57	58	59
90	5A	5B	5C	5D	5E	5F	60	61	62	63
100	64	65	66	67	68	69	6A	6B	6C	6D
110	6E	6F	70	71	72	73	74	75	76	77
120	78	79	7A	7B	7C	7D	7E	7F	80	81
130	82	83	84	85	86	87	88	89	8A	8B
140	8C	8D	8E	8F	90	91	92	93	94	95
150	96	97	98	99	9A	9B	9C	9D	9E	9F
160	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9
170	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3
180	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD
190	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7
200	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1
210	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB
220	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5
230	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
240	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
250	FA	FB	FC	FD	FE	FF	100	101	102	103
	0	1	2	3	4	5	6	7	8	9

Hexadecimális összeadótábla

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Hexadecimális szorzótábla

1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8F	9C
B	16	21	2C	37	42	4D	58	63	6E	79	84	8C	9A	A8
C	18	24	30	3C	48	54	60	6C	78	84	90	96	A5	B4
D	1A	27	34	41	4E	58	68	75	82	8F	9C	A9	B6	C3
E	1C	2A	38	46	54	62	70	7E	8C	9A	AB	B6	C4	D2
F	1E	2B	3C	48	5A	69	78	87	96	A5	B4	C3	D2	E1

----- FELJEGYZÉSEK -----

Program listák

A Függelék hátralévő részében a PRIMO periféria kezelő programjainak forrásnyelvi listái találhatók. A listákat azért közöljük, hogy az egyes rutinok használatával kapcsolatos részletkérdésekre a Z-80 gépi kódú programozásban jártas Felhasználók a programok áttanulmányozásával választ kaphassanak.

A programok listáiban látható címek mindig az adott rutin első utasításához viszonyított relatív címek. Így a rutin kezdőcímének ismeretében bármely utasítás memória címe meghatározható.

Az itt következő rutinok a PRIMO '84.1-es verziójában működnek. A PRIMO továbbfejlesztésével az alkalmazott megoldások, egy modulon belül a szubrutinok elhelyezkedése, a munkaváltozók memória címe, stb. megváltozhatnak. Annak érdekében, hogy ezek a módosítások ne befolyásolják a Felhasználók programjait, kérjük, hogy a rutinokat csak a definiált belépési ponton - ugrótábla - keresztül hívják, továbbá a modulokban található programrészeket, egyedi szubrutinokat ne használják. Az ugrótábla megkerülésével végrehajtott bármely hívás illegális, az ebből eredő problémák kizárólag a Felhasználót terhelik.

DCB = 1X = 16450 4042H

MTAB = 61 3DH

SYSTEM - PRIMO ROM START V: '84.1

```

EXT  ARAMEN,CLOCK,CLRDOT,DATUM,DCB,DIRCUR,DSPHND
EXT  ENDREC,FRERAM,GLINE,INBYTE,KKBHND,KURPOZ
EXT  KYBHND,NMIEND,PRGREC,PRTHND,PWRON,RDHEAD
EXT  RESET,SETDOT,TSTDOT,VERZIO,VJPRAM,VRST08
EXT  VRST10,VRST18,VRST20,VRST28,VRST30,VRST38
EXT  WRHEAD,WRREC
;
; BASIC ROM START
;
0000' 31                DEFB 31H                ;LD SP,--
0001'                  CHKSUM:  DEFS 2                ;16 KBYTE CHECK-SUM
0003' 18 08                JR  PWRON0
0005' C3 0000*            JPRAM:  JP  VJPRAM                ;USER JUMP
0008' C3 0000*            RST08:  JP  VRST08
000B' 0000*                DEFW FRERAM                ;SYSTEM RAM END+1
000D' C3 0000*            PWRON0: JP  PWRON                ;POWER-ON RESET
0010' C3 0000*            RST10:  JP  VRST10
0013' 0000*            DRAMST:  DEFW ARAMEN                ;DRAM NORMAL START
0015' C3 0000*            DISP:  JP  DSPHND                ;DISPLAY HANDLER
0018' C3 0000*            RST18:  JP  VRST18
001B' 0000*                DEFW DCB                ;DCB CIME
001D' C3 0000*            KYB:  JP  KYBHND                ;KEYBOARD SCANN
0020' C3 0000*            RST20:  JP  VRST20
0023' 0000*                DEFW CLOCK                ;CLOCK, STIME CIME
0025' C3 0000*            KKYB:  JP  KKBHND                ;KEYBOARD WAIT
0028' C3 0000*            RST28:  JP  VRST28
002B'                  DEFS 2
002D' C3 0000*            PRT:  JP  PRTHND                ;PRINTER HANDLER
0030' C3 0000*            RST30:  JP  VRST30
0033'                  DEFS 2
0035' C3 0000*            GETLIN: JP  GLINE                ;EGY SOR BE
0038' C3 0000*            RST38:  JP  VRST38
003B'                  DEFS 2
;
; MASZKTABLA A PONT MUVELETEKHEZ
;
003D' 80 40 20 10        MTAB:  DEFB 80H,40H,20H,10H,8,4,2,1
0041' 08 04 02 01
0045'                  DEFS 18
;
;
0057' E1                NMI2:  POP  HL                ;STATUSZ VISSZAALLITAS
0058' F1                POP  AF
0059' ED 45                RETN                ;IT ENGEDELYEZES + RET
005B' E1                NMI3:  POP  HL                ;STATUSZ TORLESE
005C' F1                POP  AF
005D' DB 03                NMI4:  IN   A,(3)
005F' E6 02                AND  2                ;RESET GOMB AKTIV MEG?

```

----- FOGGELÉK -----

```

0061' 20 FA          JR  NZ,NMI4      ;=IGEN, VARJ MEG!
0063' C3 0000*       JP  RESET        ;SYSTEM RESET
;
; NMI - NMI KISZOLGALO RUTIN
;
0066' F5             NMI:      PUSH AF      ;STATUSZ MENTES
0067' E5             PUSH HL
0068' 21 0000*       LD  HL,CLOCK      ;ORA-L BYTE CIME
006B' 34             INC  (HL)         ;CLOCK<-CLOCK+1/50 SEC.
006C' 20 07         JR  NZ,NMI1      ;=NINCS ATVITEL
006E' 2A 0001*       LD  HL,(CLOCK+1)
0071' 23             INC  HL          ;CLOCK-H,M NOVELES
0072' 22 0001*       LD  (CLOCK+1),HL
0075' DB 07         NMI1:      IN  A,(7)
0077' E6 02         AND  2           ;RESET GOMB AKTIV?
0079' CA 0000*       JP  Z,NMIEND     ;=NEM
007C' 18 DD         JR  NMI3
;
;
007E' 00*           DEFB DATUM        ;GENERALAS EVE
007F' 00*           DEFB VERZIO       ;VERZIOSZAM
;
; SYSTEM JUMP TABLA
;
0080' C3 0000*       JP  DIRCUR       ;DIREKT KURZOR CIMZES
0083' C3 0000*       JP  SETDOT       ;SET PONT
0086' C3 0000*       JP  CLRDOT       ;RESET PONT
0089' C3 0000*       JP  TSTDOT       ;TEST PONT
008C' C3 0000*       JP  KURPOZ       ;A<-AKT.KUR.POZ
008F' C3 0000*       JP  RDHEAD       ;HEADER OLVASAS
0092' C3 0000*       JP  WRHEAD       ;HEADER IRAS
0095' C3 0000*       JP  WRREC        ;BLOKK KIIRAS
0098' C3 0000*       JP  PRGREC       ;PROG.BLOKK OLVASAS
009B' C3 0000*       JP  ENDREC       ;END.BLOKK KIIRAS
009E' C3 0000*       JP  INBYTE       ;BYTE BEOLVASAS
;
; VBLANK - VARAKOZAS KEP-VISSZAFUTASRA
;
00A1' F5             VBLANK:      PUSH AF      ;STATUSZ MENTES
00A2' DB 0F         VBLK1:      IN  A,(15)
00A4' E6 20         AND  20H         ;VISSZAFUTAS?
00A6' 28 FA         JR  Z,VBLK1      ;=NEM, VARJ MEG
00A8' F1           POP  AF
00A9' C9           RET

```

----- FOGGELÉK -----

INIT - POWER-ON RESET RUTIN V: '84.1

12616

↓
0000' C9

0001' 31 0000*
0004' CD 0030'
0007' 3A 1000
000A' FE 00*
000C' C2 1000
000F' 3A 2000
0012' FE 00*
0014' C2 2000
0017' 3A 3000
001A' FE 00*
001C' C2 3000
001F' C3 0000*

0022' 2A 401F
0025' E5
0026' CD 0030'
0029' E1
002A' 22 401F
002D' C3 401E

0030' ED 56
0032' F3
0033' 21 0073'
0036' 11 4000
0039' 01 003C
003C' ED B0
003E' AF
003F' 06 44
0041' 12
0042' 13
0043' 10 FC
0045' DD 21 4042
0049' 21 1840
004C' 22 4045
004F' 3E 00*
0051' 32 404A
0054' 21 FFFF

EXT ARAMEN,C1000H,C2000H,C3000H
EXT BASIC,DSPHND,CLS,NMI2,BRESET

UTRET: RET

; PWRON - POWER-ON RESET

PWRON: LD SP,ARAMEN ;RAM TETEJE
CALL INIT ;INICIALIZALAS
CHKROM: LD A,(1000H)
CP C1000H
JP NZ,1000H ;=#1 TOK NEM BASIC
LD A,(2000H)
CP C2000H
JP NZ,2000H ;=#2 TOK NEM BASIC
LD A,(3000H)
CP C3000H
JP NZ,3000H ;=#3 TOK NEM BASIC
JP BASIC ;BASIC PWRON

; RESET - SYSTEM RESET (RESET GOMB)

RESET: LD HL,(VRESET+1)
PUSH HL ;RESET END CIME
CALL INIT ;RAM INICIALIZALAS
POP HL ;RESET END CIME
LD (VRESET+1),HL
JP VRESET ;RESET END EXEC.

; INIT - RAM TERULET INICIALIZALAS

INIT: IM 1 ;IT TIPUSA
DI ;IT TILOS
LD HL,RAMINI ;RAM KONSTANSOK
LD DE,4000H ;RAM START CIM
LD BC,60
LDIR ;UGROTABLA INIT
XOR A ;A<-0
LD B,68 ;403CH..407FH<-0
INIT1: LD (DE),A
INC DE
DJNZ INIT1 ;=VAN MEG BYTE
LD IX,DCB ;DCB CIM
LD HL,1840H
LD (DCB+3),HL
LD A,ARAMEN/256
LD (DCB+8),A ;DRAM START-H
LD HL,0FFFFH

0057'	22	404D	LD	(DCB+11),HL	
005A'	21	3A50	LD	HL,3A50H	
005D'	22	4050	LD	(DCB+14),HL;PRTHND INIT	
0060'	21	42E9	LD	HL,42E9H	
0063'	22	405A	LD	(DCB+24),HL	
0066'	3A	403B	LD	A,(PORT)	;HW PORT INIT
0069'	D3	0B	OUT	(11),A	
006B'	3E	01	LD	A,1	;DSP<-NORMAL
006D'	CD	0000*	CALL	DSPHND	
0070'	C3	0000*	JP	CLS	;DSP TORLESE
0073'	C3	1C96	JP	1C96H	;RST 08H
0076'	C3	1D78	JP	1D78H	;RST 10H
0079'	C3	1C90	JP	1C90H	;RST 18H
007C'	C3	25D9	JP	25D9H	;RST 20H
007F'	C3	0000'	JP	UTRET	;RST 28H
0082'	C3	0000'	JP	UTRET	;RST 30H
0085'	C3	0000'	JP	UTRET	;RST 38H
0088'	C3	0000'	JP	UTRET	;VJPRAM
008B'	C3	0000*	JP	NMI2	;NMIEND
008E'	C3	0000*	JP	NMI2	
0091'	C3	0000*	JP	BRESET	;VRESET
0094'	C3	0000'	JP	UTRET	
0097'	C3	0000'	JP	UTRET	
009A'	C3	0000'	JP	UTRET	;EXIT1
009D'	C3	0000	JP	0	
00A0'	C3	0000	JP	0	
00A3'	C3	0000	JP	0	
00A6'	C3	0000	JP	0	
00A9'	C3	0000	JP	0	
00AC'	0000*		DEFW	ARAMEN	;RAMEND
00AE'	ED		DEFB	OEDH	;PORT
;					
; RAM TERULET (4000H..407FH)					
;					
4000	VRST08	EQU	4000H	;RST 08H VEKTOR	
4003	VRST10	EQU	4003H	;RST 10H VEKTOR	
4006	VRST18	EQU	4006H	;RST 18H VEKTOR	
4009	VRST20	EQU	4009H	;RST 20H VEKTOR	
400C	VRST28	EQU	400CH	;RST 28H VEKTOR	
400F	VRST30	EQU	400FH	;RST 30H VEKTOR	
4012	VRST38	EQU	4012H	;RST 38H VEKTOR	
4015	VJPRAM	EQU	4015H	;USER JP VEKTOR	
4018	NMIEND	EQU	4018H	; (JP NMI2)	
00AF'		DEFS	2		
401E	VRESET	EQU	401EH	;RESET VEGE	
00B1'		DEFS	4		
4027	EXIT1	EQU	4027H	;KEYBOARD SPEC.MOD	
00B5'		DEFS	15		
;					
;					

----- F088EL&K -----

4039	RAMEND	EQU	4039H	; DRAM START CIM
403B	PORT	EQU	403BH	; SYSTEM OUTPUT PORT
				; FENNTARTVA
403D	CLOCK	EQU	403DH	; CLOCK -L, -M, -H
4042	DCB	EQU	4042H	; DEVICE CONTROL BLOCK
4062	NAMERM	EQU	4062H	; FILE-NEV PUFFER
	;			
	;			
4072	FRERAM	EQU	4072H	

DSPHND - DISPLAY HANDLER V: '84.1

belépés

21D 15H

helye 13767

3507H

```

ki EXT DCB, KARROM, KGKROM, PORT, CLRVK9, SETD1
be ENTRY DSPHND, CURNeg, CUROFF, HANG
;
; DISPLAY HANDLER
;
; 16 SOR, SORONKENT 42 KARAKTER (256*192 PONT)
; NORMAL ES ELONGALT KARAKTEREK ( E.K.=2 N.K. )
; NORMAL KARAKTER 12*6 PONTOS MEZOBEN ELHELVEZ-
; KEDO 6*8 PONT. INVERTALT ALAP ESETEN AZ 11*6
; PONT TERULETU. SULLYESZTETT KARAKTEREK 2 SOR-
; RAL LEJJEBO HELVEZKEDNEK EL. FUGGOLEGES IRAS
; ESETEN 21 SOR, SORONKENT 32 KARAKTER LEHETSE-
; GES. FUGGOLEGES IRASNAL CSAK A 10H ES 11H VE-
; ZERLO KODOK AKTIVAK. HA ILYENKOR A SOR VEGERE
; ER ( KEP FELSO SZELE ), AZ IRAST AZ ADOTT SOR
; ELEJEN FOLYTATJA.
; MEGJELENITHETO KODOK: 1EH..7FH - NORMAL ROM
; 80H..FFH - OPC. ROM/RAM
;
; VEZERLO KODOK:
; 01 - NORMAL
; 02 - ELONGACIO CTRL/B
; 03 - FEHER HATTER CTRL/C
; 04 - INVERZ ALAP CTRL/D
; 05 - UNDERLINE CTRL/E
; 06 - PRE-CLEAR CTRL/F
; 07 - BELL CTRL/G
; 08 - BACK STEP (-
; 09 - HORIZONTAL TAB SHIFT/-)
; 0A -
; 0B -
; 0C - CLEAR SCREEN CLS
; 0D - CR & LF RETURN
; 0E - ONLY CR CTRL/N
; 0F - FUGGOLEGES IRAS CTRL/O
; 10 - ALSO INDEX CTRL/P
; 11 - FELSO INDEX CTRL/Q
; 12 - ELONGACIO KI CTRL/R
; 13 - FEHER HATTER KI CTRL/S
; 14 - INVERZ ALAP KI CTRL/T
; 15 - UNDERLINE KI CTRL/U
; 16 - PRE-CLEAR KI CTRL/V
; 17 - VIZSZINTES IRAS CTRL/W
; 18...1D -
;
; REGISZTEREK:
; BE: (A)= ASCII KOD
; KI: (A), (BC), (DE), (HL)= *
; (F)= (A) SZERINT

```

----- F068ELÉK -----

		; STK:	24 BYTE	
		; DSPHND:		; REGISZTER MENTES
0000'	FD E5		PUSH IY	
0002'	E5		PUSH HL	
0003'	D5		PUSH DE	
0004'	C5		PUSH BC	
0005'	D6 1E		SUB 1EH	; 1E-)>0..7F->61
0007'	F5		PUSH AF	; MODOSITOTT KOD
0008'	21 0005*	← 16455	LD HL, DCB+5	; FLAG-BYTE CIME ← kivás vezérlo
000B'	38 3C		JR C, DSPVK	; =VEZERLO KOD
000D'	7E		LD A, (HL)	; FLAG-BYTE
000E'	E6 18	← eltolás 0000'-hoz képest	AND 18H	; PRE-CLEAR, INV.ALAP?
0010'	C4 00D8'		CALL NZ, VK084	; =IGEN, ALAP RAJZOLAS
0013'	DD CB 05 76		BIT 6, (IX+5)	; UNDERLINE? alá húzás
0017'	C4 01D1'		CALL NZ, UNDKAR	; =IGEN, RAJZOLAS
001A'	FD 21 0000*		LD IY, KARROM	; NORM.KAR.ROM CIME
001E'	F1		POP AF	; REL.KAR.ROM CIM
001F'	F5		PUSH AF	
0020'	FE 62		CP 62H	; FELHASZNALOI KAR.?
0022'	38 06		JR C, DSP1	; =1E..7F, NORM.ROM
0024'	D6 62		SUB 62H	; 80->0..FF->7F
0026'	FD 2A 0009*		LD IY, (DCB+9)	; OPC.KAR.ROM CIME
002A'	26 00	DSP1:	LD H, 0	; *ROM CIM GENERALAS*
002C'	87		ADD A, A	
002D'	6F		LD L, A	; HL<-2*REL.CIM
002E'	29		ADD HL, HL	
002F'	29		ADD HL, HL	; (HL)=8*REL.CIM
0030'	EB		EX DE, HL	
0031'	FD 19		ADD IY, DE	; (IY)=KAR.ROM CIM
0033'	CD 01D5'		CALL KARGEN	; KAR. RAJZOLAS
0036'	ED 53 0006*		LD (DCB+6), DE	; SOR/OSZLOP-CIM MENT
003A'	1C		INC E	
003B'	1D		DEC E	; ELERT A SOR VEGERE?
003C'	CC 014A'		CALL Z, VK0D	; =IGEN, CR(ROLL)
003F'	F1	DSP2:	POP AF	; MODOSITOTT KOD
0040'	C6 1E		ADD A, 1EH	; (A)=EREDETI KOD
0042'	B7		OR A	; CY<-0 (PICTURE)
0043'	C1		POP BC	; REGISZTER VISSZAMENT
0044'	D1		POP DE	
0045'	E1		POP HL	
0046'	FD E1		POP IY	
0048'	C9		RET	; RET DSPHND
0049'	C6 1E	DSPVK:	ADD A, 1EH	; EREDETI KOD
004B'	47		LD B, A	; SZLO KEZDOERTEK
004C'	11 003F'		LD DE, DSP2	; FOLYTATASI CIM
004F'	D5		PUSH DE	
0050'	10 15		DJNZ VK02-2	; =NEM 01 (NORMAL)
0052'	CD 0269'	VK01:	CALL CUOFF	; KURZOR KIKAPCSOLASA
0055'	21 0005*		LD HL, DCB+5	; FLAG-BYTE CIME
0058'	CD 01A6'		CALL VK13	; FEHER HATTER KI

↓
vezérlo
karakter-
ek

----- FOGGELÉK -----

005B' 36 00		LD (HL),0	;FLAG-BYTE(-ALAPERTEK
005D' 2B		DEC HL	;KLAVIATURA FLAG-BYTE
005E' CB 96		RES 2,(HL)	;KURZOR FLAG(-0
0060' 21 0000*		LD HL,KGKROM	;80H FELETT A KGKROM
0063' 22 0009*		LD (DCB+9),HL	;MUKODJON
0066' C9		RET	
0067' 10 03		DJNZ VK03-2	;=NEM 02 (ELONGACIO BE)
0069' CB FE	VK02:	SET 7,(HL)	;ELONGACIO BE
006B' C9		RET	
006C' 10 23		DJNZ VK04-2	;=NEM 03 (FEHER HATTER)
006E' CB 6E	VK03:	BIT 5,(HL)	;FEHER A HATTER?
0070' C0		RET NZ	;=MAR FEHER
0071' CB EE		SET 5,(HL)	;FEHER HATTER BE
0073' 01 0018	VK031:	LD BC,24	;6144=24*256
0076' DD 66 08		LD H,(IX+8)	;DRAM-START-H
0079' 68		LD L,B	
007A' 7E	VK032:	LD A,(HL)	;DRAM KOMPLEMENT
007B' 2F		CPL	
007C' 77		LD (HL),A	
007D' 23		INC HL	;POINTER ELORE
007E' 10 FA		DJNZ VK032	;=VAN MEG
0080' 0D		DEC C	
0081' 20 F7		JR NZ,VK032	;=VAN MEG
0083' 21 0005*	VK033:	LD HL,DCB+5	;FLAG-BYTE CIME
0086' 7E		LD A,(HL)	;HA A FEHER HATTER ES
0087' 17		RLA	;AZ INVERZ ALAP FLAG
0088' AE		XOR (HL)	;ERTEKE ELTER, AKKOR
0089' E6 20		AND 20H	;INVERZ RAJZ FLAG(-1,
008B' CB 8E		RES 1,(HL)	;EGYEBKENT FLAG(-0
008D' CB		RET Z	;=NORMAL RAJZOLAS
008E' CB CE		SET 1,(HL)	;INVERZ RAJZOLAS
0090' C9		RET	
0091' 10 07		DJNZ VK05-2	;=NEM 04 (INVERZ ALAP)
0093' CB 66	VK04:	BIT 4,(HL)	;INVERZ ALAP?
0095' C0		RET NZ	;=MAR INVERZ
0096' CB E6		SET 4,(HL)	;INVERZ ALAP BE
0098' 18 E9		JR VK033	;INVERZ RAJZ F. BEALL.
009A' 10 03		DJNZ VK06-2	;=NEM 05 (UNDERLINE)
009C' CB F6	VK05:	SET 6,(HL)	;UNDERLINE BE
009E' C9		RET	
009F' 10 05		DJNZ VK07-2	;=NEM 06 (PRE-CLEAR)
00A1' CB DE	VK06:	SET 3,(HL)	;PRE-CLEAR(-1
00A3' CB D6		SET 2,(HL)	;PRE-CLEAR-PUF.(-1
00A5' C9		RET	
00A6' 10 11		DJNZ VK08-2	;=NEM 07 (BELL)
00A8' 11 7500	VK07:	LD DE,7500H	;800 HZ, 1/6 SEC
00AB' 3A 0000*	HANG:	LD A,(PORT)	;PORT TARTALOM BYTE
00AE' 42	VK071:	LD B,D	;IDOZITES
00AF' EE 10		XOR 10H	;DUDA(-NOT(DUDA)
00B1' D3 03		OUT (3),A	

00B3'	10	FE	
00B5'	1D		
00B6'	20	F6	
00B8'	C9		
00B9'	10	44	
00BB'	ED	5B 0006*	VK08:
00BF'	01	060C	
00C2'	7B		
00C3'	18	05	
00C5'	7A		VK081:
00C6'	91		
00C7'	DB		
00C8'	57		
00C9'	AF		
00CA'	CB	7E	VK082:
00CC'	2B	01	
00CE'	90		
00CF'	90		VK083:
00D0'	BB		
00D1'	5F		
00D2'	30	F1	
00D4'	ED	53 0006*	
00D8'	FD	21 025E'	VK084:
00DC'	7E		VK085:
00DD'	F5		
00DE'	30	0E	
00E0'	F6	02	
00E2'	CB	6F	
00E4'	2B	02	
00E6'	EE	02	
00E8'	DD	CB 03 56	VK086:
00EC'	2B	02	
00EE'	EE	02	VK087:
00F0'	77		VK088:
00F1'	CD	01D5'	
00F4'	F1		
00F5'	32	0005*	
00F8'	C9		

```

DJNZ $ ;VARAKOZAS
DEC E ;1/6 SEC LETELT?
JR NZ,VK071 ;=NEM, DUDALJ MEG!
RET
DJNZ VK09-2 ;=NEM 08 (BACK STEP)
LD DE,(DCB+6) ;SOR/OSZLOP CIM
LD BC,060CH ;B<-6, C<-12
LD A,E ;OSZLOPCIM
JR VK082
LD A,D ;SORCIM
SUB C ;EGY SORRAL FEL
RET C ;=KEP ELEJEN ALL
LD D,A ;UJ SORCIM MENT
XOR A ;OSZLOPCIM<-256
BIT 7,(HL)
JR Z,VK083 ;=NINCS ELONGACIO
SUB B ;OSZLOP<-OSZLOP-12
SUB B ;OSZLOP<-OSZLOP-6
CP E ;BAL SZELEN TUL?
LD E,A ;UJ OSZLOPCIM
JR NC,VK081 ;=IGEN, ELOZO SORBA
LD (DCB+6),DE ;SOR/OSZLOP CIM MENT
LD IY,KARBST ;BSTEP KAR.ROM CIM
LD A,(HL) ;FLAG-BYTE MENT
PUSH AF
JR NC,VK087 ;=PRE-CLEAR, ALAP RAJZ
OR 2 ;FLAG<-INVERZ RAJZ
BIT 5,A
JR Z,VK086 ;=FEKETE HATTER
XOR 2 ;FLAG<-NORMAL RAJZ
BIT 2,(IX+3)
JR Z,VK088 ;=KURZOR INAKTIV
XOR 2 ;FLAG<-NOT(INV.RAJZ)
LD (HL),A ;FLAG<-UJ ERTEKEK
CALL KARGEN ;BSTEP/UNDERLINE RAJZ
POP AF
LD (DCB+5),A ;FLAG<-EREDETI ERTEK
RET

```

: H-TAB POZICIO TABLA (TERM. A DJNZ)

00F9'	04 34 64 94	TABTAB:	DEFB 4,52,100,148,196,244
00FD'	C4 F4		
00FF'	10 25		DJNZ VK0C-4 ;=NEM 09 (H-TAB)
0101'	23	VK09:	INC HL ;OSZLOPCIM CIME
0102'	5E		LD E,(HL) ;OSZLOPCIM
0103'	DD 56 07	VK091:	LD D,(IX+7) ;SORCIM
0106'	06 0B		LD B,11 ;TORLENDŐ PONTSZAM
0108'	CD 0000*	VK092:	CALL CLRVK9 ;TORLES/BEALLITAS
010B'	14		INC D ;SORCIM NOVELES

----- FÜGGELÉK -----

010C' 3E BF		LD A,191	
010E' BA		CP D	;KEPERNYO ALJA?
010F' 38 02		JR C,VK093	;=IGEN, KOV.OSZLOP
0111' 10 F5		DJNZ VK092	;=VAN MEG PONT
0113' 1C	VK093:	INC E	;OSZLOPCIM NOVELES
0114' 28 34		JR Z,VK0D	;=JOBB SZEL, CR
0116' 21 00F8'		LD HL,TABTAB-1	;H-TAB POZ.TABLA
0119' 23	VK094:	INC HL	
011A' 7E		LD A,(HL)	;KOV.TABLAELEM
011B' CB 57		BIT 2,A	;TERMINATOR?
011D' 28 E4		JR Z,VK091	;=IGEN, UJ OSZLOPOT
011F' BB		CP E	;H-TAB POZICIO?
0120' 20 F7		JR NZ,VK094	;=NEM, KOVETKEZOT
0122' DD 73 06		LD (IX+6),E	;OSZLOPCIM MENT
0125' C9		RET	
0126' 05		DEC B	;0A, 0B KOD NINCS
0127' 05		DEC B	
0128' 10 1E		DJNZ VK0D-2	;=NEM 0C (CLEAR SCREEN)
012A' CD 01C4'	VK0C:	CALL VK16	;PRE-CLEAR KI
012D' AF		XOR A	;A<-0
012E' 21 0104		LD HL,0104H	;SOR<-1, OSZLOP<-4
0131' 22 0006*		LD (DCB+6),HL	
0134' 6F		LD L,A	;L<-0
0135' 5C		LD E,H	;E<-1
0136' DD 66 08		LD H,(IX+8)	; (HL)=DRAM-START
0139' 54		LD D,H	; (DE)=DRAM-START+1
013A' 01 17FF		LD BC,6143	;DRAM MERET-1
013D' DD CB 05 6E	VK0C1:	BIT 5,(IX+5)	;FEHER HATTER?
0141' 28 01		JR Z,VK0C2	;=FEKETE, KOD<-0
0143' 3D		DEC A	;FEHER, KOD<-FFH
0144' 77	VK0C2:	LD (HL),A	;DRAM FELTOLTESE A
0145' ED B0		LDIR	;MEGFEELELO KODDAL
0147' C9		RET	
0148' 10 24		DJNZ VK0E-2	;=NEM 0D (CR & LF)
014A' 21 0006*	VK0D:	LD HL,DCB+6	;OSZLOPCIM CIME
014D' 36 04		LD (HL),4	;OSZLOPCIM<-4
014F' 23		INC HL	;SORCIM CIME
0150' 3E 0C		LD A,12	;1 BETUSOR=12 KEPSOR
0152' 86		ADD A,(HL)	;SORCIM<-KOV.SOR CIME
0153' FE C0		CP 192	;KEP ALJAN TUL?
0155' 30 02		JR NC,VK0D1	;=IGEN, ROLL
0157' 77		LD (HL),A	;UJ SORCIM MENT
0158' C9		RET	
0159' AF	VK0D1:	XOR A	;A<-0 ← Display RAM
015A' 23		INC HL	;DRAM-START-H CIME
015B' 56		LD D,(HL)	;DRAM-START-H
015C' 5F		LD E,A	; (DE)=DRAM-START
015D' 62		LD H,D	;H<-DRAM-START
015E' 24		INC H	;H<-DRAM-START+256
015F' 2E 80		LD L,128	; (HL)=DRAM-START+384

----- FOGGELÉK -----

0161' 01 1680		LD BC,5760	;ELTOLANDO BYTE-SZAM
0164' ED B0		LDIR	;ROLL FEL
0166' 62		LD H,D	
0167' 6B		LD L,E	; (HL)=TOLTENDO TER.CIM
0168' 1C		INC E	; (DE)=(HL)+1
0169' 01 017F		LD BC,383	;FELTOLTENDO BYTE-SZAM
016C' 18 CF		JR VK0C1	;UTOLSO SOR FELTOLTESE
016E' 10 04		DJNZ VK0F-2	;=NEM OE (ONLY CR)
0170' 23	VK0E:	INC HL	;OSZLOPCIM CIME
0171' 36 04		LD (HL),4	;OSZLOPCIM(-4
0173' C9		RET	
0174' 10 03		DJNZ VK10-2	;=NEM OF (FUGG. IRAS)
0176' CB C6	VK0F:	SET 0,(HL)	;IRANY(-FUGGOLEGES
0178' C9		RET	
0179' 10 1D		DJNZ VK11-2	;=NEM 10 (ALSO INDEX)
017B' 3E 04	VK10:	LD A,4	;4 KEPSORRAL LE
017D' CB 46	VK101:	BIT 0,(HL)	
017F' 2A 0006*		LD HL,(DCB+6)	;SOR/OSZLOP CIM
0182' 20 0C		JR NZ,VK104	;=FUGGOLEGES IRAS
0184' 84		ADD A,H	;SOR(-SOR (+-) 4
0185' 67		LD H,A	;UJ SORCIM MENT
0186' 05		DEC B	
0187' 28 02		JR Z,VK102	;=11 (FELSO INDEX)
0189' FE C0		CP 192	;MAX. SORCIM
018B' D0	VK102:	RET NC	;=KEPEN KIVULRE, NEM!
018C' 22 0006*	VK103:	LD (DCB+6),HL	;SOR/OSZLOP CIM MENT
018F' C9		RET	
0190' 85	VK104:	ADD A,L	;OSZLOP(-OSZLOP(+-)4
0191' 6F		LD L,A	;UJ OSZLOPCIM MENT
0192' 05		DEC B	
0193' 28 F6		JR Z,VK102	;=11 (FELSO INDEX)
0195' D8		RET C	;=KEPEN KIVULRE, NEM!
0196' 18 F4		JR VK103	;SOR/OSZLOP CIM MENT
0198' 10 05		DJNZ VK12-2	;=NEM 11 (FELSO INDEX)
019A' 04	VK11:	INC B	;B(-1
019B' 3E FC		LD A,-4	;4 KEPSORRAL FEL
019D' 18 DE		JR VK101	
019F' 10 03		DJNZ VK13-2	;=NEM 12 (ELONGACIO KI)
01A1' CB BE	VK12:	RES 7,(HL)	;ELONGACIO KI
01A3' C9		RET	
01A4' 10 08		DJNZ VK14-2	;=NEM 13 (SOTET HATTER)
01A6' CB 6E	VK13:	BIT 5,(HL)	;FEKETE A HATTER?
01A8' CB		RET Z	;=MAR FEKETE
01A9' CB AE		RES 5,(HL)	;HATTER FLAG(-FEKETE
01AB' C3 0073'		JP VK031	;KEP KOMPLEMENTALASA
01AE' 10 0D		DJNZ VK15-2	;=NEM 14 (NORMAL ALAP)
01B0' 7E	VK14:	LD A,(HL)	;FLAG-BYTE
01B1' E6 E7		AND OE7H	;INV.ALAP(-PRE-CLEAR(-0
01B3' CB 57		BIT 2,A	;PRE-CLEAR-PUF.=1?
01B5' 28 02		JR Z,VK141	;=NEM, PRE-CLEAR(-0

----- FOGGELÉK -----

01B7' F6 08		OR 8	;PRE-CLEAR(-1
01B9' 77	VK141:	LD (HL),A	;FLAG-BYTE MENT
01BA' C3 0083'		JP VK033	
01BD' 10 03		DJNZ VK16-2	;=NEM 15 (UNDERLINE KI)
01BF' CB B6	VK15:	RES 6,(HL)	;UNDERLINE KI
01C1' C9		RET	
01C2' 10 08		DJNZ VK17-2	;=NEM 16 (PRE-CLEAR KI)
01C4' CB 96	VK16:	RES 2,(HL)	;PRE-CLEAR-PUF(-0
01C6' CB 66		BIT 4,(HL)	;INVERZ ALAP?
01C8' C0		RET NZ	;=IGEN, PRE-CLEAR MARAD
01C9' CB 9E		RES 3,(HL)	;PRE-CLEAR KI
01CB' C9		RET	
01CC' 10 02		DJNZ VK17+2	;=NEM 17 (VIZSZ. IRAS)
01CE' CB 86	VK17:	RES 0,(HL)	;IRANY(-VIZSZINTES
01D0' C9		RET	;18..1D KOD NINCS

;
 ; UNDKAR - UNDERLINE KIRAJZOLASA

01D1' FD 21 0257'	UNDKAR:	LD IY,KARUND	;UNDERLINE KAR.ROM
-------------------	---------	--------------	--------------------

;
 ; KARGEN - KARAKTER KIRAJZOLO RUTIN

01D5' ED 5B 0006*	KARGEN:	LD DE,(DCB+6)	;SOR/OSZLOP CIM
01D9' DD CB 05 46		BIT 0,(IX+5)	
01DD' 20 38		JR NZ,KGENY	;=FUGGOLEGES IRAS
01DF' FD CB 00 7E	KGENX:	BIT 7,(IY+0)	;*VIZSZINTES IRAS*
01E3' 28 07		JR Z,KGENX2	;=NINCS SULLYESZTES
01E5' 14		INC D	;2 SORRAL LE
01E6' 14	KGENX1:	INC D	;KOV. SORBA
01E7' 3E BF		LD A,191	;KEP ALJA
01E9' BA		CP D	
01EA' 38 27		JR C,KGENX7	;=KEP ALJAN VAN, KESZ
01EC' 06 06	KGENX2:	LD B,6	;KAR. SZELESSEGE
01EE' DD 5E 06		LD E,(IX+6)	;EREDETI OSZLOPCIM
01F1' FD 7E 00		LD A,(IY+0)	;KOV. KAR. SOR
01F4' 17		RLA	;SULLYESZTES-BIT KI
01F5' 87	KGENX3:	ADD A,A	;CY(-KOV. PONT
01F6' DD CB 05 7E		BIT 7,(IX+5)	
01FA' 28 06		JR Z,KGENX4	;=NEM ELONGALT
01FC' DC 0000*		CALL C,SETD1	;AKTIV PONT RAJZOLAS
01FF' 1C		INC E	;KOV. OSZLOPBA
0200' 28 09		JR Z,KGENX6	;=KEP JOBB SZELE
0202' DC 0000*	KGENX4:	CALL C,SETD1	;AKTIV PONT RAJZOLAS
0205' 1C		INC E	;KOV. OSZLOPBA
0206' 28 03		JR Z,KGENX6	;=KEP JOBB SZELE
0208' 10 EB		DJNZ KGENX3	;=VAN MEG PONT
020A' 17	KGENX5:	RLA	;CY(-TERMINATOR
020B' 05	KGENX6:	DEC B	;SOR ELNYELES?
020C' F2 020A'		JP P,KGENX5	;=IGEN, TOVABB
020F' FD 23		INC IY	;KAR.KOV.SORARA

F84.0.

----- FOGGELAK -----

0211'	30 D3		JR	NC,KGENX1	;=VAN MEG KAR.SOR
0213'	DD 56 07	KGENX7:	LD	D,(IX+7)	;EREDETI SORCIM
0216'	C9		RET		
0217'	14	KGENY:	INC	D	;*FUGGOLEGES IRAS*
0218'	15		DEC	D	;SOR VEGEN ALL?
0219'	20 05		JR	NZ,KGENY0	;=NINCS A SOR VEGEN
021B'	16 BF		LD	D,191	;SOR<-191, SOR ELEJERE
021D'	DD 72 07		LD	(IX+7),D	
0220'	FD CB 00 7E	KGENY0:	BIT	7,(IY+0)	
0224'	28 06		JR	Z,KGENY2	;=NINCS SULLYESZTES
0226'	1C		INC	E	;2 OSZLOPPAL JOBBRA
0227'	28 2A		JR	Z,KGENY7	;=KEP JOBB SZELE, KESZ
0229'	1C	KGENY1:	INC	E	;KOV. OSZLOPBA
022A'	28 27		JR	Z,KGENY7	;=KEP JOBB SZELE, KESZ
022C'	06 06	KGENY2:	LD	B,6	;KAR. SZELESSEGE
022E'	DD 56 07		LD	D,(IX+7)	;EREDETI SORCIM
0231'	FD 7E 00		LD	A,(IY+0)	;KOV. KAR. SOR
0234'	17		RLA		;SULLYESZTES-BIT KI
0235'	87	KGENY3:	ADD	A,A	;CY<-KOV. PONT
0236'	DD CB 05 7E		BIT	7,(IX+5)	
023A'	28 06		JR	Z,KGENY4	;=NEM ELONGALT
023C'	DC 0000*		CALL	C,SETD1	;AKTIV PONT RAJZOLAS
023F'	15		DEC	D	;ELOZO SORBA
0240'	28 09		JR	Z,KGENY6	;=KEP TETEJE
0242'	DC 0000*	KGENY4:	CALL	C,SETD1	;AKTIV PONT RAJZOLAS
0245'	15		DEC	D	;ELOZO SORBA
0246'	28 03		JR	Z,KGENY6	;=KEP TETEJE
0248'	10 EB		DJNZ	KGENY3	;=VAN MEG PONT
024A'	17	KGENY5:	RLA		;CY<-TERMINATOR
024B'	05	KGENY6:	DEC	B	;SOR ELNYELES?
024C'	F2 024A'		JP	P,KGENY5	;=IGEN, TOVABB
024F'	FD 23		INC	IY	;KAR.KOV.SORARA
0251'	30 D6		JR	NC,KGENY1	;=VAN MEG KAR.SOR
0253'	DD 5E 06	KGENY7:	LD	E,(IX+6)	;EREDETI OSZLOPCIM
0256'	C9		RET		
					; UNDERLINE & BACK-STEP KAR. ROM
0257'	80 80 80 80	KARUND:	DEFB	80H,80H,80H,80H,80H,80H,0FFH	
025B'	80 80 FF				
025E'	7E FE FE FE	KARBST:	DEFB	7EH,0FEH,0FEH,0FEH,0FEH,0FEH	
0262'	FE FE				
0264'	FE FE FE FE		DEFB	0FEH,0FEH,0FEH,0FEH,0FFH	
0268'	FF				
					; CUROFF, CURNEG - KURZOR RAJZOLO RUTINOK
0269'	DD CB 03 56	CUROFF:	BIT	2,(IX+3)	
026D'	C8		RET	Z	;=KURZOR INAKTIV
026E'	3A 0003*	CURNEG:	LD	A,(DCB+3)	;KURZOR BIT INVERTALAS

----- F066EL&K -----

0271' EE 04
0273' 32 0003*
0276' FD 21 0257'
027A' 21 0005*
027D' 37
027E' C3 00DC'

XOR 4
LD (DCB+3),A
LD IY,KARUND ;KURZOR=UNDERLINE
LD HL,DCB+5 ;DSP FLAG-BYTE CIM
SCF ;(CY)=KURZOR RAJZOLAS
JP VK085 ;RAJZOLAS + RET

----- FOGSÉLEK -----

KYBHND - BILLENTYUZET HANDLER V:'84.1

```

EXT    DCB,HANG,CUROFF,CURNEG
ENTRY  KYBHND,KKBHND,BREAK,GOMBQ
;
;      KEYBOARD HANDLER
;
;  MEGVIZSGALJA, HOGY A BILLENTYUZETEN NYOMJAK-E
;  VALAMELYIK BILLENTYUT, ES HA TALAL AKTIV GOM-
;  BOT, ANNAK KODJAT AZ A REGISZTERBEN ADJA MEG.
;  VALAMELY GOMB SOKSZOROS LEKERDEZESET CSAK AB-
;  BAN AZ ESETBEN KEZDI MEG, HA AZ ADOTT GOMB A
;  GYORS SCANNELES ALATT AKTIVNAK BIZONYULT. EK-
;  KOR 128-SZOR MEGVIZSGALJA, MAJD AKKOR TEKINTI
;  AKTIVNAK, HA A SOKSZOROS LEKERDEZES ALATT LE-
;  GALABB EGYSZER AKTIV VOLT. A VEZERLO GOMBOKAT
;  - SHIFT, CNT, UPPER - MINDEN ESETBEN SOKSZO-
;  ROS LEKERDEZESSEL VIZSGALJA. HA A DCB-BEN TA-
;  ROLT KEYBOARD-FLAG-BYTE 1-ES BITJE = 1, BREAK
;  KODOT NEM KULD. HA A 0-AS BIT = 1, SPECIALIS
;  UZEMMODBAN MUKODIK A HANDLER. EKKOR A LENYIL
;  BILLENTYU VEZERLO GOMBKENT - EXTRA SHIFT - U-
;  ZEMEL. A LENYILLAL EGYUTT AKTIV GOMB KODJA AZ
;  EREDETI KODNAL 80H-VAL NAGYOBB LESZ.
;  A DCB+4 BYTE TAROLJA A SOKSZOROS LEKERDEZESOK
;  SORAN KET VIZSGALAT KOZOTTI IDOZITES IDOTAR-
;  TAMAT. EZ ALAPERTELMEZESBEN 160 MIKROSEC (24)
;  EZT AZ ERTEKET PROGRAMBOL MEGVALTOZTATVA LAS-
;  SITANI, ILL. GYORSITANI LEHET A KLAVIATURAT.
;  AZ IDOZITO MIN ERTEKE = 10, MAX ERTEKE = 255.
;
;  REGISZTEREK:
;  BE:          NINCS
;  KI:          (BC),(DE),(HL)= *
;              (A)= ASCII KOD
;              (F)= (A) SZERINT
;  STK:         10 BYTE
;
;  KYBHND:      PUSH HL          ;REGISZTER MENTES
;              PUSH DE
;              PUSH BC
;              XOR  A
;              LD  E,A          ;FELTETELEZETT UPPER=0
;              LD  B,A          ;INIT KLAV10 SZAMARA
;              ADD A,(IX+2)      ;AKTIV GOMB CIME
;              JP  P,KLAV1      ;=UPPER = 0
;              SET 7,E          ;UPPER(-1
;
;  KLAV1:      BIT 6,A
;              JR  NZ,KLAV2      ;=NEM VOLT AKTIV GOMB
;              AND 3FH          ;AKTIV GOMB CIME
;
0000' E5
0001' D5
0002' C5
0003' AF
0004' 5F
0005' 47
0006' DD 86 02
0009' F2 000E'
000C' CB FB
000E' CB 77
0010' 20 0F
0012' E6 3F

```

----- FOGGELÉK -----

0014' FE 0F		CP 0FH	; LENYIL VOLT AKTIV?
0016' 28 09		JR Z, KLAV2	; =IGEN, NEM AKTIV!
0018' 4F		LD C, A	; GOMB CIM A HELYERE
0019' CD 00BB'		CALL GOMBM	; MOST IS AKTIV?
001C' 3A 0000*		LD A, (DCB+0)	; AKTIV ASCII KODJA
001F' 38 56		JR C, KLAV71	; =IGEN, KOD KESZ!
0021' 0E 40	KLAV2:	LD C, 40H	; UTOLSO GOMB CIM + 1
0023' 0D	KLAV3:	DEC C	
0024' FA 00A6'		JP M, KLAV10	; =NINCS AKTIV GOMB
0027' 79		LD A, C	; KOVETKEZO GOMB CIM
0028' FE 03		CP 3	
002A' 28 F7		JR Z, KLAV3	; =SHIFT, NE NEZD!
002C' FE 07		CP 7	
002E' 28 F3		JR Z, KLAV3	; =CTRL, NE NEZD!
0030' FE 0F		CP 0FH	
0032' 20 06		JR NZ, KLAV4	; =NEM A LENYIL
0034' DD CB 03 46		BIT 0, (IX+3)	
0038' 20 E9		JR NZ, KLAV3	; =SPEC. MOD, NE NEZD!
003A' CD 00B7'	KLAV4:	CALL GOMBQ	; AKTIV EZ?
003D' 30 E4		JR NC, KLAV3	; =NEM, A KOVETKEZOT
003F' 79		LD A, C	; AKTIV GOMB CIME
0040' FE 05		CP 5	
0042' 28 53		JR Z, KLAV9	; =UPPER AZ AKTIV
0044' B3		OR E	; A7(-UPPER BIT
0045' 32 0002*		LD (DCB+2), A	; UPPER & AKTIV CIM MENT
0048' CB 39	KLAV5:	SRL C	; GOMB CIM/2
004A' 30 0F		JR NC, KLAV6	; =BETU (PAROS), (B)=0
004C' CB 21		SLA C	; CIM VISSZAALLITAS
004E' 21 0125'		LD HL, KLAVT2-3	; NEM-BETU KODTABLA-3
0051' 09		ADD HL, BC	; NEM-SHIFTES CIME-3
0052' 0E 03		LD C, 3	
0054' CD 00BB'		CALL GOMBM	; CY(-1, HA SHIFT AKTIV
0057' ED 4A		ADC HL, BC	; (HL)=A KOD CIME
0059' 18 1A		JR KLAV7	
005B' 21 0108'	KLAV6:	LD HL, KLAVT1	; BETU KODTABLA
005E' 09		ADD HL, BC	; (HL)=ALAPKOD CIME
005F' 0E 07		LD C, 7	
0061' CD 00BB'		CALL GOMBM	; CTRL AKTIV?
0064' 38 0F		JR C, KLAV7	; =IGEN, KOD(-ALAPKOD
0066' CB F3		SET 6, E	; KOD(-ALAPKOD+40H
0068' CB 7B		BIT 7, E	; UPPER?
006A' 20 09		JR NZ, KLAV7	; =IGEN, KOD(-KOD+40H
006C' 0E 03		LD C, 3	
006E' CD 00BB'		CALL GOMBM	; SHIFT AKTIV?
0071' 38 02		JR C, KLAV7	; =IGEN, KOD(-KOD+40H
0073' CB EB		SET 5, E	; KOD(-ALAPKOD+60H
0075' 7E	KLAV7:	LD A, (HL)	; AZ ALAPKOD
0076' B3		OR E	; MODOSITOTT KOD
0077' E6 7F	KLAV71:	AND 7FH	; AZ ASCII KOD
0079' DD CB 03 46		BIT 0, (IX+3)	

----- FOGGELÉK -----

007D'	28 09		JR	Z, KLAV8	;=NEM SPEC. MOD
007F'	17		RLA		;RRA MIATT
0080'	67		LD	H, A	;KOD MENTES
0081'	0E 0F		LD	C, 0FH	
0083'	CD 00BB'		CALL	GOMBM	;CY<-1, HA LENYIL AKTIV
0086'	7C		LD	A, H	;KOD VISSZAMENTES
0087'	1F		RRA		;KOD<-KOD+80H, HA AKTIV
0088'	32 0000*	KLAV8:	LD	(DCB+0), A	;BILLENTYU KOD MENTESE
008B'	FE 01		CP	1	
008D'	20 77		JR	NZ, KKLAV5	;=NEM BREAK, RET
008F'	DD CB 03 4E		BIT	1, (IX+3)	
0093'	28 71		JR	Z, KKLAV5	;=NINCS BREAK TILTAS
0095'	18 0F		JR	KLAV10	;KOD<-0, RET
0097'	63	KLAV9:	LD	H, E	;UPPER MENT
0098'	11 F02B		LD	DE, 0F02BH	;400 HZ, 1/10 SEC
009B'	CD 0000*		CALL	HANG	;BILLENTYU HANG
009E'	5C		LD	E, H	
009F'	CD 00BB'		CALL	GOMBM	;UPPER GOMB AKTIV?
00A2'	38 FB		JR	C, \$-3	;=IGEN, VARJ MEG!
00A4'	06 80		LD	B, 80H	;UPPER<-NOT(UPPER)
00A6'	78	KLAV10:	LD	A, B	;A<-0, VAGY 80H
00A7'	AB		XOR	E	;A7<-UPPER BIT
00A8'	F6 40		OR	40H	;NINCS AKTIV JELZESE
00AA'	32 0002*		LD	(DCB+2), A	;UPPER & AKTIV CIM MENT
00AD'	AF		XOR	A	
00AE'	18 DB		JR	KLAV8	;KOD<-0, RET
;					
; BREAK - BREAK GOMB VIZSGALATA					
;					
; MEGVIZSGALJA, HOGY A BREAK GOMB AKTIV-E. HA					
; BREAK TILTAS VAN, VIZSGALAT NELKUL INAKTIV					
; ALLAPOTTAL TER VISSZA.					
;					
; REGISZTEREK:					
; BE: NINCS					
; KI; (E), (HL)= *					
; (A), (B), (D)= ?					
; (C)=3FH					
; (CY)= 1, AKTIV A BREAK					
; STK: 2 BYTE					
;					
00B0'	DD CB 03 4E	BREAK:	BIT	1, (IX+3)	;BREAK TILTAS?
00B4'	C0		RET	NZ	;=IGEN, BREAK INAKTIV
00B5'	0E 3F		LD	C, 3FH	;3FH=BREAK
;					
; GOMBQ - GOMB GYORS LEKERDEZESE					
;					
00B7'	ED 78	GOMBQ:	IN	A, (C)	;GOMB LEKERDEZES
00B9'	1F		RRA		;CY<-ERTEK
00BA'	D0		RET	NC	;=INAKTIV

```

;
; GOMB1 - GOMB SOKSZOROS LEKERDEZESE
;
00BB' 16 7F      GOMB1:      LD      D,127      ;127=128-1, SZLO
00BD' 06 80      LD      B,128      ;LEKERDEZESEK SZAMA
00BF' 3A 0004*   GOMB1:      LD      A,(DCB+4)  ;MIN 60 MIKROSEC DELAY
00C2' 3D          DEC      A
00C3' 20 FD      JR      NZ,$-1      ;=VARJ MEG!
00C5' ED 78      IN      A,(C)      ;GOMB LEKERDEZES
00C7' E6 01      AND      1          ;A0<-ERTEK
00C9' 82          ADD      A,D      ;SZLO<-SZLO+ERTEK
00CA' 57          LD      D,A
00CB' 10 F2      DJNZ     GOMB1      ;=TOVABBI LEKERDEZES
00CD' 17          RLA              ;CY<-1, HA SZLO>128
00CE' C9          RET              ;(CY)=1, HA AKTIV
;
; KIVARASOS KLAVIATURA HANDLER
;
; MUKODESE SORAN A KLAVIATURAT ADDIG SCANNELI,
; AMIG AZON AKTIV BILLENTYUT NEM TALAL. EZ IDO
; ALATT A KURZOR VILLOG. FOLYAMATOS NYOMAS HA-
; TASARA - EGY IDOZITES LETELTE UTAN - AUTOMA-
; TIKUS ISMETLO FUNKCIO INDUL MEG.
;
; REGISZTEREK:
; BE:          NINCS
; KI:          (BC),(DE),(HL)= *
;              (A)= ASCII KOD
;              (F)= (A) SZERINT
; STK:         24 BYTE
;
00CF' E5      KKBHND:  PUSH HL      ;REGISZTER MENTES
00D0' D5      PUSH DE
00D1' C5      PUSH BC
00D2' FD E5   PUSH IY
00D4' DD 56 00 LD      D,(IX+0)    ;ELOZO KOD
00D7' D5      KKLAV1:  PUSH DE      ;ELOZO MENTES
00D8' CD 0000* CALL CURNEG ;KURZOR NEGALAS
00DB' D1      POP DE
00DC' 1E 3C   LD      E,KLAVK1    ;VILLOGAS IDO
00DE' 3E      DEFB 3EH            ;LD A,-
00DF' 57      KKLAV2:  LD      D,A  ;AKT.KOD=NULL
00E0' 1D      KKLAV3:  DEC      E  ;VILLOGTATAS?
00E1' 28 F4   JR      Z,KKLAV1    ;=IGEN, NEGALAS
00E3' CD 0000* CALL KYBHND ;KLAV. SCANNELES
00E6' 28 F7   JR      Z,KKLAV2    ;=KLAV. URES!
00E8' 92      SUB      D          ;=ELOZOVEL AZONOS?
00E9' 20 08   JR      NZ,KKLAV4    ;=NEM AZ ELOZO
00EB' DD 35 01 DEC      (IX+1)    ;ISMETLES LEJART?
00EE' 20 F0   JR      NZ,KKLAV3    ;=NEM, UJRA

```

----- FOGGELÉK -----

00F0' 3E 01		LD A,KLAVK2	;ROVID IDOZITES
00F2' 21		DEFB 21H	;LD HL,--
00F3' 3E 0F	KKLAV4:	LD A,KLAVK3	;HOSSZU IDOZITES
00F5' 32 0001*		LD (DCB+1),A	;IDOZITES INIT
00F8' CD 0000*		CALL CUROFF	;KURZOR KIKAPCS.
00FB' 11 F028		LD DE,0F028H	;400 HZ, 1/10 SEC
00FE' CD 0000*		CALL HANG	;BILL.HANG
0101' 3A 0000*		LD A,(DCB+0)	;A KOD
0104' FD E1		POP IY	
0106' 18 22	KKLAV5:	JR KLAV11	;REG. VISSZA, RET
003C	KLAVK1	EQU 60	;VILLOGAS SZLO.
0001	KLAVK2	EQU 1	;ROVID IDOZITES
000F	KLAVK3	EQU 15	;HOSSZU IDOZITES
0108' 19	KLAVT1:	DEFB 019H	;00 00 Y
0109' 13		DEFB 013H	;01 02 S
010A' 05		DEFB 005H	;02 04 E
010B' 17		DEFB 017H	;03 06 W
010C' 04		DEFB 004H	;04 08 D
010D' 18		DEFB 018H	;05 0A X
010E' 11		DEFB 011H	;06 0C Q
010F' 01		DEFB 001H	;07 0E A
0110' 03		DEFB 003H	;08 10 C
0111' 06		DEFB 006H	;09 12 F
0112' 12		DEFB 012H	;0A 14 R
0113' 14		DEFB 014H	;0B 16 T
0114' 08		DEFB 008H	;0C 18 H
0115' 02		DEFB 002H	;0D 1A B
0116' 07		DEFB 007H	;0E 1C G
0117' 16		DEFB 016H	;0F 1E V
0118' 0E		DEFB 00EH	;10 20 N
0119' 1A		DEFB 01AH	;11 22 Z
011A' 15		DEFB 015H	;12 24 U
011B' 0A		DEFB 00AH	;13 26 J
011C' 0C		DEFB 00CH	;14 28 L
011D' 0B		DEFB 00BH	;15 2A K
011E' 0D		DEFB 00DH	;16 2C M
011F' 09		DEFB 009H	;17 2E I
0120' 1E		DEFB 01EH	;18 30 UE
0121' 10		DEFB 010H	;19 32 P
0122' 0F		DEFB 00FH	;1A 34 O
0123' 00		DEFB 0	;1B 36 NINCS
0124' 00		DEFB 0	;1C 38 NINCS
0125' 00		DEFB 000H	;1D 3A EE
0126' 1D		DEFB 01DH	;1E 3C AE
0127' 1C		DEFB 01CH	;1F 3E OE
0128' 1F	KLAVT2:	DEFB 01FH	
0129' 1F		DEFB 01FH	;00 01 FNYIL

----- FOGGELÉK -----

```

;
; KLA111 - KYBHND, KKBHND TERMINALAS
;
012A' B7      KLA111:      OR      A      ;FLAG-EK A KOD SZERINT
012B' C1      POP      BC      ;REG. VISSZAMENTES
012C' D1      POP      DE
012D' E1      POP      HL
012E' C9      RET          ;RET KYBHND, KKBHND
;
012F' 00      DEFB 0      ;NINCS
0130' 33      DEFB 033H   ;      3
0131' 23      DEFB 023H   ;04 09 &
0132' 32      DEFB 032H   ;      2
0133' 22      DEFB 022H   ;05 0B "
0134' 31      DEFB 031H   ;      1
0135' 21      DEFB 021H   ;06 0D !
0136' 0A      DEFB 00AH   ;
0137' 1A      DEFB 01AH   ;07 0F LNYIL
0138' 00      DEFB 0      ;
0139' 00      DEFB 0      ;08 11 NINCS
013A' 00      DEFB 0      ;
013B' 00      DEFB 0      ;09 13 NINCS
013C' 00      DEFB 0      ;
013D' 00      DEFB 0      ;0A 15 NINCS
013E' 37      DEFB 037H   ;      7
013F' 2F      DEFB 02FH   ;0B 17 /
0140' 20      DEFB 020H   ;
0141' 20      DEFB 020H   ;0C 19 SPACE
0142' 36      DEFB 036H   ;      6
0143' 26      DEFB 026H   ;0D 1B &
0144' 35      DEFB 035H   ;      5
0145' 25      DEFB 025H   ;0E 1D x
0146' 34      DEFB 034H   ;      4
0147' 24      DEFB 024H   ;0F 1F $
0148' 38      DEFB 038H   ;      8
0149' 28      DEFB 028H   ;10 21 (
014A' 2B      DEFB 02BH   ;      +
014B' 3F      DEFB 03FH   ;11 23 ?
014C' 30      DEFB 030H   ;      0
014D' 3D      DEFB 03DH   ;12 25 =
014E' 3E      DEFB 03EH   ;      )
014F' 3C      DEFB 03CH   ;13 27 (
0150' 2D      DEFB 02DH   ;      -
0151' 1E      DEFB 01EH   ;14 29 HOSSZU KIS I
0152' 2E      DEFB 02EH   ;      .
0153' 3A      DEFB 03AH   ;15 2B :
0154' 39      DEFB 039H   ;      9
0155' 29      DEFB 029H   ;16 2D )
0156' 2C      DEFB 02CH   ;      ,
0157' 3B      DEFB 03BH   ;17 2F ;

```

----- FOGGEL&K -----

0158' 2A	DEFB 02AH	; *
0159' 27	DEFB 027H	;18 31 '
015A' 5F	DEFB 05FH	; HOSSZU KIS U
015B' 7F	DEFB 07FH	;19 33 HOSSZU KIS UE
015C' 0C	DEFB 00CH	
015D' 0C	DEFB 00CH	;1A 35 CLS
015E' 0D	DEFB 00DH	
015F' 0D	DEFB 00DH	;1B 37 CR
0160' 0B	DEFB 008H	
0161' 18	DEFB 018H	;1C 39 BNYIL
0162' 5B	DEFB 05BH	; HOSSZU KIS O
0163' 7B	DEFB 07BH	;1D 3B HOSSZU KIS OE
0164' 19	DEFB 019H	
0165' 09	DEFB 009H	;1E 3D JNYIL
0166' 01	DEFB 001H	
0167' 01	DEFB 001H	;1F 3F BREAK

PRTHND - PRINTER HANDLER V: '84.1

```

EXT    DCB,DISP
ENTRY PRTHND
;
; PRINTER HANDLER
;
; AZ A REGISZTERBEN KAPOTT KARAKERT A PRIN-
; TERRE LISTAZZA, MAJD MEGVIZSGALJA, HOGY A
; SOR HOSSZA ELERTE-E A DCB-BEN TAROLT MAXI-
; MALIS SORHOSSZAT. HA NEM, RETURN. HA IGEN,
; A PRINTERRE EGY CR KODOT IR, MAJD MEGVIZS-
; GALJA, HOGY BETELT-E A LAP. HA BETELT, AK-
; KOR A PRINTERRE EGY FF KODOT IR, HACSAK A
; DCB-BEN TAROLT LAPMERET NEM NAGYOBB, MINT
; 127. HA NAGYOBB, AKKOR A FORM FEED VEZERLO
; KOD SOSE KERUL KIIRASRA.
; VALAMELY NEM VEZERLOKOD KARAKTER KINYOMTA-
; TASA UTAN A PRINTER KURZOR ERTEKE 1-EL NO.
; AMENNYIBEN NEM TILTJUK, A HANDLER KET KO-
; DOT ATKONVERTAL (1EH->69H, 1FH->5EH).
;
; REGISZTEREK:
; BE:      (A)= ASCII KOD
; KI:      (A),(BC),(DE),(HL)= *
;          (F)= (A) SZERINT
; STK:     8 BYTE
;
PRTHND:  OR    A
0000' B7      RET    Z           ;=BLANK
0001' C8      PUSH   BC         ;REGISZTER MENTES
0002' C5      PUSH   AF         ;KAR.+FLAG MENT
0003' F5      BIT    3,(IX+3)
0004' DD CB 03 5E  JR    NZ,PRTNK ;=KONVERZIO TILOS
0008' 20 0C    CP     1EH
000A' FE 1E    JR    NZ,PRTK1    ;=NEM HOSSZU KIS I
000C' 20 02    LD     A,69H      ;A<-ROVID KIS I
000E' 3E 69    CP     1FH
0010' FE 1F    PRTK1:  JR    NZ,PRTNK ;=NEM FNYIL
0012' 20 02    LD     A,5EH      ;A<-ASCII FNYIL
0014' 3E 5E    PRTNK:  LD     C,A  ;KAR. MENTES
0016' 4F      CP     ' '
0017' FE 20    JR     C,PRTVK    ;=VEZERLO KOD
0019' 3B 0D    CALL   PRTOUT     ;KARAKTER PRINT
001B' CD 0064' INC    (IX+18)    ;PRINTER KURZOR ELORE
001E' DD 34 12 DEC    (IX+16)    ;KAR.SZLO CSOKKENT
0021' DD 35 10 JR     NZ,PRT2    ;=NEM SORVEG
0024' 20 36    LD     A,0DH      ;A<-CR
0026' 3E 0D    PRTVK:  CP     0CH
0028' FE 0C    JR     Z,PRTFF    ;=FORM FEED
002A' 2B 0D

```

```

002C' FE 0D                CP      0DH
002E' 2B 04                JR      Z,PRTCR      ;=CARRIGE RET
0030' FE 0A                CP      0AH
0032' 20 25                JR      NZ,PRT1      ;=NEM LINE FEED
0034' DD 35 11             PRTCR:  DEC      (IX+17)      ;SORSZLO CSOKKENT
0037' 20 14                JR      NZ,PRTCR1     ;=NEM LAPVEG
0039' 3A 000F*             PRTFF:  LD      A,(DCB+15)      ;LAPHOSSZ
003C' B7                   OR      A
003D' FA 004D'             JP      M,PRTCR1     ;=255, FORM FEED NE!
0040' 32 0011*             LD      (DCB+17),A      ;SORSZLO(-LAPHOSSZ
0043' 0E 0C                LD      C,0CH
0045' CD 0064'             CALL    PRTOUT      ;FORM FEED KI
0048' 0E 0D                LD      C,0DH
004A' CD 0064'             CALL    PRTOUT      ;KET URES SOR!
004D' 0E 0D             PRTCR1:  LD      C,0DH      ;A(-CR
004F' 3A 000E*             LD      A,(DCB+14)      ;KAR.SZLO(-SORHOSSZ
0052' 32 0010*             LD      (DCB+16),A
0055' DD 36 12 00          LD      (IX+18),0      ;PRINTER KURZOR INIT
0059' CD 0064'             PRT1:   CALL    PRTOUT      ;KARAKTER PRINT
005C' F1                   PRT2:   POP      AF      ;KAR. + FLAG
005D' C1                   POP      BC      ;REG. VISSZAMENTES
005E' C9                   RET
005F' 3E 07             PROUT1:  LD      A,7      ;HANGJELZES
0061' CD 0000*             CALL    DISP

;
; PRTOUT - EGY KARAKTER A PRINTERRE
;
0064' DB 05             PRTOUT:  IN      A,(5)      ;STATUSZ BE
0066' CB 67             BIT      4,A
0068' 20 F5             JR      NZ,PROUT1     ;=OFF LINE
006A' E6 08             AND      B
006C' 20 F6             JR      NZ,PRTOUT     ;=BUSY, MEGINT
006E' 79             LD      A,C
006F' D3 40             OUT      (100Q),A      ;KAR. A PRINTERRE
0071' C9             RET

```

GLINE - EGY SORT BEOLVASÓ RUTIN V:'84.1

```

EXT  DISP,KKYB,EXIT1
ENTRY GLINE
;
; GLINE - EGY SOR A KEYBOARD BUFFERBE
;
; BEOLVAS A KEYBOARDROL ES MEGJELENIT A DISPLAY
; AKTUALIS SORATOL KEZDODOEN EGY MAX. 210 DARAB
; KARAKTERBOL ALLO SORT. A KARAKTEREKET BEIRJA
; A (HL) ALTAL KIJELOLT KEYBOARD PUFFERBE.
; A CR, BREAK, H-TAB, SHIFT/(-, CLS ES BACKSTEP
; VEZERLO KODOKTOL KULONBOZO 1EH-NAL KISEBB KO-
; DOKAT CSAK ECHO-ZZA, DE AZOKRA NEM REAGAL, ES
; - A TOBBI VEZERLO KODHOZ HASONLOAN - NEM IRJA
; BE EZEKET A KEYBOARD PUFFERBE. HA MAR 210 KA-
; RAKTERT ELTAROLT, AZ ESETLEGES TOVABBBI KARAK-
; TEREK BEERKEZESEKOR HANGJELZEST AD, DE A KA-
; RAKTERT MAR NEM TAROLJA ES NEM ECHO-ZZA.
; CR ES BREAK KOD HATASARA A PUFFER VEGERE EGY
; 0-AT IR, MAJD A VEZERLEST VISSZAADJA AZ AKTI-
; VIZALO RUTINNAK. A H-TAB KODOT ECHOZAS NELKUL
; ELNYELI. BSTEP HATASARA A PUFFER JELENLEGI U-
; TOLSO KARAKTERET TORLI, MIG SHIFT/(- HATASARA
; A TELJES SORT TORLI. UGYANEZT TESZI A CLS HA-
; TASARA IS.
; A RUTIN A KLAVIATURAT SPECIALIS MODBAN HIVJA.
; HA 80H-NAL NAGYOBB KOD ERKEZIK, A VEZERLEST A
; A RUTIN AZ EXIT1 PONTRA ADJA, A KOVETKEZO RE-
; GISZTER TARTALMAKKAL:
; HL - PUFFER POINTER
; B - SOR AKTUALIS HOSSZA
; C - SOR LEHETSEGES HOSSZA
; A - KARAKTERKOD (>80H)
; AZ EXIT1 PONTON - HA SZUKSEGES - A DE REGISZ-
; TERPART MENTENI KELL!
;
; REGISZTEREK:
; BE: (HL)= PUFFER CIM - 1
; KI: (BC),(DE),(HL)= *
; (A)= 0DH
; (CY)= 1, BREAK-ET NYOMTAK
; STK: 30 BYTE
;
GLINE:  PUSH BC          ;REGISZTER MENTES
        XOR  A           ;NULL (GL3 MIATT)
        DEFB 06H         ;LD B,-
GL1:    POP  HL          ;PUFFER KEZDOCIM
        PUSH HL          ;KEZDOCIM MENT
        LD   B,210       ;A SOR MAX HOSSZA

```

```

0000' C5
0001' AF
0002' 06
0003' E1
0004' E5
0005' 06 D2

```

----- FOGGEL&K -----

0007' 48		LD C,B	
0008' 23	GL2:	INC HL	; PUFFER POINTER ELORE
0009' CD 0000*	GL3:	CALL DISP	; KARAKTER ECHO
000C' 3E		DEFB 3EH	; LD A,-
000D' F1	GL4:	POP AF	; 1 SZO TORLES
000E' DD CB 03 C6	GL5:	SET 0,(IX+3)	; KEYB.SPEC.MOD BE
0012' CD 0000*		CALL KKYB	; BILLENTYU KOD BE
0015' DD CB 03 86		RES 0,(IX+3)	; SPEC.MOD KI
0019' FE 80		CP 80H	; SPEC. KOD?
001B' D4 0000*		CALL NC,EXIT1	; =IGEN, BASIC EXIT #1
001E' FE 1E		CP 1EH	
0020' 38 06		JR C,GL6	; =VEZ.KOD
0022' 77		LD (HL),A	; KOD A PUFFERBE
0023' 10 E3		DJNZ GL2	; =NINCS MEG TELE A SOR
0025' 04		INC B	; B(-1
0026' 3E 07		LD A,7	; HIBA!, HANGJELZES
0028' FE 0D	GL6:	CP 0DH	
002A' 28 27		JR Z,GLCR	; =CR, SORZARAS
002C' FE 02		CP 2	
002E' 38 21		JR C,GLBRK	; =BREAK, SORZARAS
0030' FE 09		CP 9	
0032' 28 DA	GL7:	JR Z,GL5	; =H-TAB, ELNYELI
0034' FE 18		CP 18H	
0036' 28 08		JR Z,GLBSTP	; =SHIFT/(-, SORTORLES
0038' FE 0C		CP 0CH	
003A' 28 C7		JR Z,GL1	; =CLS, SOR ELOROL
003C' FE 08		CP 8	
003E' 20 C9		JR NZ,GL3	; =NEM BSTEP, ECHO+UJ BE
0040' F5	GLBSTP:	PUSH AF	; 08H, 18H MENT
0041' 79		LD A,C	
0042' B8		CP B	; URES MEG/MAR A PUFFER?
0043' 28 C8		JR Z,GL4	; =IGEN, SOR ELOROL
0045' 04		INC B	; SORHOSSZ SZLO-1
0046' 2B		DEC HL	; POINTER VISSZA
0047' 3E 08		LD A,8	
0049' CD 0000*		CALL DISP	; BSTEP A DSP-RE
004C' F1		POP AF	; 8, VAGY 18H
004D' FE 08		CP 8	
004F' 18 E1		JR GL7	; =8->GL5, =18H->GLBSTP
0051' 3E 0D	GLBRK:	LD A,0DH	
0053' F5	GLCR:	PUSH AF	; (CY)=1, HA BREAK
0054' CD 0000*		CALL DISP	; KURZOR A KOV. SORBA
0057' 36 00		LD (HL),0	; TERMINATOR A PUFFERBE
0059' F1		POP AF	; (A)=0DH, BREAK, (CY)=1
005A' E1		POP HL	; PUFFER CIM-1
005B' C1		POP BC	; MENTETT BC
005C' C9		RET	

BEEP - ELEMI HANGGENERATOR V: '84.1

```

EXT    PORT
ENTRY BEEP
;
; BEEP - ELEMI HANGGENERATOR
;
; A RUTIN A MEGADOTT HOSSZUSAGU, ES PERIODUS I-
; DEJU HANGOT GENERALJA A BELSO HANGSZORON ES A
; MAGNO FESZULTSEGGIMENETEN.
; HOSSZUSAG (T2) = 2*T1*(BC)    MICROSEC
; PERIODUS/2 (T1) = 8.4*(DE)+35  MIKROSEC
;
; REGISZTEREK:
; BE:      (BC)= HANG HOSSZUSAG
;          (DE)= PERIODUSIDO / 2
; KI:      (DE),(HL)= *
;          (BC)= 0FFFFH
;          (AF)= ?
; STK:     8 BYTE
;
0000' FD E5      BEEP:    PUSH IY
0002' E5         PUSH HL          ; MENTES
0003' C5         PUSH BC          ; IY<-HOSSZUSAG
0004' FD E1      POP  IY
0006' 01 FFFF    LD   BC,-1
0009' 3A 0000*   LD   A,(PORT)    ; SYSTEM OUTPUT BYTE
000C' E6 FC      AND  0FCH        ; MAGNO<-'0'
000E' CD 0021'   BEEP1:  CALL BEEP2    ; FELSO FELPERIODUS
0011' CD 0021'   CALL BEEP2    ; ALSO FELPERIODUS
0014' FD 09      ADD  IY,BC        ; <-HOSSZUSAG-1
0016' DA 000E'   JP   C,BEEP1     ; =HOSSZUSAG<)0
0019' F6 02      OR   2           ; MAGNO<-'GND'
001B' D3 06      OUT  (6),A
001D' E1         POP  HL          ; MENTETT ERTEK
001E' FD E1      POP  IY
0020' C9         RET
0021' EE 13      BEEP2:  XOR  13H    ; NOT(DUDA), NOT(MAGNO)
0023' D3 06      OUT  (6),A
0025' 62         LD   H,D          ; HL<-PERIODUS / 2
0026' 6B         LD   L,E
0027' 09         BEEP3:  ADD  HL,BC   ; <-PERIODUS-1
0028' DA 0027'   JP   C,BEEP3     ; =PERIODUS<)0
002B' C9         RET

```

KURPOZ - KURZOR POZÍCIÓT MEGÁLLAPÍTÓ RUTIN V: '84.1

```

EXT DCB
;
; KURPOZ - DSP KURZORCIM MEGHATÁROZÁSA
;
; MEGÁLLAPÍTJA A KURZOR SORON BELÜLI AKTUALIS
; POZÍCIÓJÁT, ÉS EZT AZ ÉRTEKET KILEPESKÖR AZ
; A REGISZTERBE TESZI. HA A KURZOR POZÍCIÓJA
; NEM SZABVÁNYOS OSZLOP/SOR ÉRTEK, KILEPESKÖR
; A CIM EZEN ÉRTEKHEZ LEGKÖZELEBB ESŐ, DE NA-
; LA NAGYOB B ÉRTEK LESZ. VÍZSZINTES ÍRAS ESE-
; TEN, HA OSZLOPCIM<4, A POZÍCIÓ =0 LESZ.
;
; REGISZTEREK:
; BE:      NINCS
; KI:      (BC),(DE),(HL)= *
;          (A)= KURZOR POZÍCIÓ
;          (F)= ?
; STK:     4 BYTE
;
0000' E5      KURPOZ:  PUSH HL          ;REGISZTER MENTES
0001' 21 0005* LD    HL,DCB+5        ;DSP FLAG-BYTE CÍME
0004' CB 46    BIT    0,(HL)
0006' 23      INC    HL
0007' 20 0F    JR     NZ,KURPO4      ;=FÜGGŐLEGES ÍRAS
0009' 7E      LD     A,(HL)          ;DSP OSZLOPCIM
000A' D6 04    SUB    4              ;ELTOLÁS KI
000C' 21 FF06  KURPO1: LD    HL,OFF06H ;H<- -1, L<-6
000F' 38 01    JR     C,KURPO3      ;=OSZLOPCIM<4
0011' 95      KURPO2: SUB    L        ;6-AL CSÖKKENTÉS
0012' 24      KURPO3: INC    H        ;SZLO NOVELESE
0013' 30 FC    JR     NC,KURPO2     ;=LE LEHET MEG VONNI
0015' 7C      LD     A,H            ;KURPOZ<-SZÁMLÁLÓ
0016' E1      POP    HL             ;MENTETT HL
0017' C9      RET
0018' 23      KURPO4: INC    HL
0019' 3E BF    LD     A,191          ;MAX SORCIM
001B' 96      SUB    (HL)           ;DSP SORCÍME
001C' 18 EE    JR     KURPO1

```

DIRCUR - DIREKT KURZOR CIMZÉS RUTIN V:'84.1

```

EXT   DCB
;
;   DIRCUR - DIREKT KURZOR CIMZES SZUBRUTIN
;
;   A REGISZTEREKBEN MEGADOTT SOR/OSZLOP CIMRE
;   ALLITJA A DISPLAY KURZORT.
;   HA A MEGADOTT SOR/OSZLOP SZAM TUL NAGY, A
;   VEZERLEST - HIBAJELZESSEL - VISSZAKAPJA A
;   HIVO RUTIN. ILYEN ESETBEN A KURZOR POZICI-
;   OJA NEM VALTOZIK.
;
;   REGISZTEREK:
;   BE:      (D)= SOR-SZ      V.=0..15, F.=0..20
;            (E)= OSZLOP-SZ V.=0..41, F.=0..31
;   KI:      (BC),(DE),(HL)= *
;            (A)= ?
;            (CY)= C, TUL NAGY SOR/OSZLOP SZAM
;   STK:      4 BYTE
;
0000' 3E 14   DIRCUR:  LD  A,20      ;MAX SOR-SZ
0002' BA      CP  D
0003' D8      RET  C      ;=TUL NAGY SOR-SZ
0004' 3E 29   LD  A,41     ;MAX OSZLOP-SZ
0006' BB      CP  E
0007' D8      RET  C      ;=TUL NAGY OSZLOP-SZ
0008' E5      PUSH HL     ;REGISZTER MENT
0009' 7A      LD  A,D      ;SORCIM<-S-SZ*12+1
000A' 87      ADD  A,A
000B' 82      ADD  A,D      ;3*S-SZ
000C' 87      ADD  A,A
000D' 87      ADD  A,A      ;12*S-SZ
000E' 3C      INC  A
000F' 67      LD  H,A      ;SORCIM MENT
0010' 7B      LD  A,E      ;OSZLOPCIM<-O-SZ*6+4
0011' 87      ADD  A,A
0012' 83      ADD  A,E      ;3*O-SZ
0013' 87      ADD  A,A
0014' C6 04   ADD  A,4      ;6*O-SZ+4
0016' 6F      LD  L,A      ;OSZLOPCIM MENT
0017' DD CB 05 46  BIT  0,(IX+5)
001B' 20 0A   JR  NZ,DIRCU3 ;=FUGGOLEGES IRAS
001D' 3E BF   LD  A,191     ;MAX SORCIM
001F' BC      CP  H
0020' 38 03   JR  C,DIRCU2  ;=TUL NAGY SORSZAM
0022' 22 0006* DIRCU1: LD  (DCB+6),HL ;SOR/OSZLOPCIM MENT
0025' E1      DIRCU2: POP  HL  ;REGISZTER VISSZA
0026' C9      RET
0027' 3E C3   DIRCU3: LD  A,195 ;MAX SORCIM+4

```

----- FOGGELÉK -----

0029'	95	SUB	L	
002A'	38 F9	JR	C, DIRCU2	;=TUL NAGY OSZLOPSZAM
002C'	24	INC	H	;SORCIM<-S-SZ*12+4
002D'	24	INC	H	
002E'	24	INC	H	
002F'	6C	LD	L,H	;OSZLOPCIM<-SORCIM
0030'	67	LD	H,A	;SORCIM<-OSZLOPCIM
0031'	18 EF	JR	DIRCU1	

PONT - ELEMI PONT MOVELETEK V: '84.1

Leírás 11-660

```

EXT DCB,MTAB
;
; SETDOT - GRAFIKUS PONT BEKAPCSOLASA
;
; A RUTIN BEKAPCSOLJA A MEGADOTT KOORDINATAJU
; GRAFIKUS PONTOT A KEPERNYO AKTUALIS ALAPSZI-
; NENEK MEGFELELOEN. A KURZOR NEM MOZDUL EL.
; HA AZ ADOTT PONT A KEPERNYON KIVULRE ESNE, A
; VEZERLEST - HIBAJELZES KISERETEBEN - A HIVO
; RUTIN VISSZAKAPJA ES RAJZOLAS NEM TORTENIK.
;
; REGISZTEREK:
; BE:      (DE)= ABSZOLUT KOORDINATAK
; KI:      (DE)= KONVERTALT KOORDINATAK
;          (A),(B),(HL)= *
;          (C)= ? (BIT MASZK)
;          (CY)= C, HIBAS Y-KOORDINATA
; STK:     10 BYTE
;
15734
↓
0000' 3E BF   SETDOT:  LD  A,191      ;MAX Y-KOORD
0002' 92      SUB  D              ;0->191, 191->0
0003' D8      RET  C              ;=TUL NAGY Y-KOORD
0004' 57      LD  D,A            ;D<-KONVERTALT Y-KOORD
0005' E5      SETD01:  PUSH HL     ;REGISZTER MENT
0006' CD 0029' CALL SETD1        ;PONT BEKAPCSOLAS
0009' E1      POP  HL
000A' C9      RET

;
; CLRDOT - GRAFIKUS PONT KIKAPCSOLASA
;
; A RUTIN KIKAPCSOLJA A MEGADOTT KOORDINATAJU
; GRAFIKUS PONTOT A KEPERNYO AKTUALIS ALAPSZI-
; NENEK MEGFELELOEN. A KURZOR NEM MOZDUL EL.
; HA AZ ADOTT PONT A KEPERNYON KIVULRE ESNE, A
; VEZERLEST - HIBAJELZES KISERETEBEN - A HIVO
; RUTIN VISSZAKAPJA ES RAJZOLAS NEM TORTENIK.
;
; REGISZTEREK:
; BE:      (DE)= ABSZOLUT KOORDINATAK
; KI:      (DE)= KONVERTALT KOORDINATAK
;          (A),(B),(HL)= *
;          (C)= ? (BIT MASZK)
;          (CY)= C, HIBAS Y-KOORDINATA
; STK:     10 BYTE
;
15759
↓
000B' 3E BF   CLRDOT:  LD  A,191    ;MAX Y-KOORD
000D' 92      SUB  D              ;0->191, 191->0
000E' D8      RET  C              ;=TUL NAGY Y-KOORD

```

----- FOGGELÉK -----

```
000F' 57          LD  D,A          ;D<-KONVERTALT Y-KOORD
0010' E5          CLRDO1:  PUSH HL      ;REGISZTER MENT
0011' CD 0037'    CALL CLRDO1      ;PONT KIKAPCSOLAS
0014' E1          POP  HL
0015' C9          RET
```

```
;
; TSTDOT - GRAFIKUS PONT TESZTELESE
;
; MEGVIZSGALJA A DE REGISZTER-PAR ALTAL CIMZETT
; PONT ALLAPOTAT ES ANNAK MEGFELELOEN BEALLITJA
; A Z-FLAG ERTEKET.
; HA AZ ADOTT PONT A KEPERNYON KIVULRE ESNE, A
; VEZERLEST - HIBAJELZESSEL - A HIVO RUTIN KAP-
; JA VISSZA. ILYENKOR A Z-FLAG DEFINIALATLAN.
```

```
;
; REGISZTEREK:
; BE:      (DE)= ABSZOLUT KOORDINATAK
; KI:      (DE)= KONVERTALT KOORDINATAK
;          (B)= *
;          (A),(HL)= ?
;          (C)= ? (BIT MASZK)
;          (CY)= C, HIBAS Y-KOORDINATA
;          (Z)= NZ, A PONT AKTIV
; STK:     4 BYTE
```

16167

↓

```
0016' 3E BF      TSTDOT:  LD  A,191      ;MAX Y-KOORD
0018' 92          SUB  D          ;0->191, 191->0
0019' D8          RET  C          ;=TUL NAGY Y-KOORD
001A' 57          LD  D,A          ;D<-KONVERTALT Y-KOORD
001B' CD 0046'    TSTD1:  CALL CIMGEN      ;C<-MASZK, HL<-CIM
001E' DD CB 05 6E  BIT  5,(IX+5)      ;INVERZ HATTER?
0022' 20 02       JR  NZ,TSTD2      ;=INVERTALT RAJZ
0024' A6          AND  (HL)         ;Z<-NZ, HA VILAGOS
0025' C9          RET
0026' A6          TSTD2:  AND  (HL)
0027' A9          XOR  C          ;Z<-NZ, HA SOTET
0028' C9          RET
```

```
;
; SETD1 - GRAFIKUS PONT BEKAPCSOLASA
```

```
;
; SETD1:  PUSH AF
002A' CD 0046'    SETD2:  CALL CIMGEN      ;C<-MASZK, HL<-CIM
002D' DD CB 05 4E  SETD3:  BIT  1,(IX+5)
0031' 20 0E       JR  NZ,CLRD4      ;=INVERTALT RAJZ
0033' B6          SETD4:  OR   (HL)         ;PONT BEKAPCSOLAS
0034' 77          LD  (HL),A        ;RAM<-PONT
0035' F1          POP  AF
0036' C9          RET
```

```
;
; CLRDO1 - GRAFIKUS PONT KIKAPCSOLASA
```

----- FOGGELÉK -----

```

;
0037' F5          CLRD1:    PUSH AF
0038' CD 0046'     CLRD2:    CALL CIMGEN      ;C<-MASZK, HL<-CIM
003B' DD CB 05 4E CLRD3:    BIT 1,(IX+5)
003F' 20 F2          JR     NZ,SETD4      ;=INVERTALT RAJZ
0041' 2F          CLRD4:    CPL              ;A<-KOMPLEMENS MASZK
0042' A6          AND     (HL)            ;PONT KIKAPCSOLAS
0043' 77          LD      (HL),A          ;RAM<-PONT
0044' F1          POP     AF
0045' C9          RET

;
; CIMGEN - RAM-CIM GENERATOR
;
16186
↓
0046' 21 0000*     CIMGEN:   LD      HL,MTAB      ;MASZKTABLA (LAPHATAR!)
0049' 7B          LD      A,E
004A' E6 07        AND      7              ;BIT CIM
004C' 85          ADD     A,L
004D' 6F          LD      L,A              ;HL<-MASZK CIME
004E' 4E          LD      C,(HL)          ;C<-MASZK
004F' 3A 0008*     LD      A,(DCB+8)       ;RAM-START-H
0052' 6B          LD      L,E              ;L<-OSZLOPCIM
0053' 62          LD      H,D              ;H<-SORCIM
0054' CB 3C        SRL     H              ;A SOR/OSZLOP CIMBOL
0056' CB 1D        RR      L              ;KEPZETT 16 BITES ERTE-
0058' CB 3C        SRL     H              ;KET 8-AL OSZTVA MEG-
005A' CB 1D        RR      L              ;JUK A KARESETT PONTOT
005C' CB 3C        SRL     H              ;TARTALMAZO BYTE RELA-
005E' CB 1D        RR      L              ;TIV RAM-CIMET
0060' 84          ADD     A,H
0061' 67          LD      H,A              ;HL<-BYTE ABSZ. CIME
0062' 79          LD      A,C              ;A<-MASZK
0063' C9          RET

```

MAGNO - MAGNO KEZELŐ RUTINOK V: '84.1

helye:
149M

```

EXT   DIRCUR, DISP, PORT, NAMERM, DCB
ENTRY WRHEAD, ENDREC, WRSYN, MTRON, MTROFF, WRREC, WRREC1
ENTRY RDHEAD, CHKSUM, PRGREC, RDSYN, INBYTE, CASDSP
;
;   MAGNETOFON HANDLER
;
;   A MAGNETOFON HANDLER EGYES RUTINJAI A FILE-
;   OK KÜLÖNBÖZŐ RESZEINEK (HEADER, PROGRAM ES
;   ADAT BLOKK, END BLOKK) FELÍRASAT ES VISSZA-
;   TÖLTÉSET VÉGEZIK.
;
;   MAGNETOFON IDŐALAPOK:
;
002E   TETA2      EQU      46      ;2*IDŐALAP
008A   TETA6      EQU      138     ;6*IDŐALAP
0021   TETA15     EQU      33      ;1.5*IDŐALAP
;
;   WRREC - PROGRAM ES ADAT BLOKK FELÍRASA
;
;   REGISZTEREK:
;   BE:      (BC)=ELTOLÁSI CIM /OFFSZET/
;            (DE)=MEMORIATERÜLET KEZDŐCIME
;            (HL)=MEMORIATERÜLET VÉG CIME
;            (DCB+26)=A BLOKK TÍPUSA
;            (DCB+28)=A BLOKK SORSZÁMA
;   KI:      (BC)=*
;            (DE)=0
;            (HL)=MEMORIATERÜLET VÉG CIME
;            (A)=?
;            (DCB+26)=*
;            (DCB+28)=A BLOKK SORSZÁMA+1
;   STK:      14 BYTE
;
0000' AF   WRREC:   XOR    A          ;BLOKK FELVÉTELE, CY<-0
0001' ED 52   SBC    HL, DE          ;(HL)=BYTEOK SZÁMA
0003' EB     EX    DE, HL           ;(DE)=BYTEOK SZÁMA
0004' C8     RET    Z              ;=BYTEOK SZÁMA NULLA
0005' AF   WRREC1: XOR    A          ;CY<-0
0006' E5     PUSH   HL             ;REGISZTER MENTÉSE
0007' C5     PUSH   BC
0008' ED 42   SBC    HL, BC          ;OFFSZET CIM KÉPZÉSE
000A' CD 009D' CALL   WRSYN          ;BLOKK SZINKRON FELÍRÁS
000D' 3A 001A* LD     A, (DCB+26)    ;(A)=BLOKK TÍPUSA
0010' CD 00B4' CALL   OBYTE          ;BLOKK TÍPUS FELÍRASA
0013' 3A 001C* LD     A, (DCB+28)    ;(A)=BLOKK SORSZÁMA
0016' C6 01   ADD    A, 1           ;BLOKK SORSZÁMA+1
0018' 27     DAA                   ;BLOKK SORSZÁMA BCD !
0019' 4F     LD     C, A           ;CHECKSUM=BLOKK SORSZ.

```

001A' 32 001C*	LD (DCB+28),A ;	UJ BLOKK SORSZ. MENTES
001D' CD 00B4'	CALL OBYTE ;	BLOKK SORSZ. FELIRASA
0020' 7D	LD A,L ;	MEMORIATER. KEZDOCIM
0021' CD 00B0'	CALL WBYTE ;	ALSO BYTE FELIRASA
0024' 7C	LD A,H ;	MEMORIATER. KEZDOCIM
0025' CD 00B0'	CALL WBYTE ;	FELSO BYTE FELIRASA
0028' 7B	LD A,E ;	A BLOKKBAN LEVO BYTEDK
0029' CD 00B0'	CALL WBYTE ;	DARABSZAMANAK FELIRASA
002C' E1	POP HL ;	REG. VISSZAMENTES
002D' E3	EX (SP),HL ;	(HL)=MEMORIA CIME
002E' 43	LD B,E ;	(B)=BYTEDK SZAMA
002F' 7E	LD A,(HL)	AKTUALIS BYTE BEVETELE
0030' 23	INC HL ;	(HL)=KOV. MEMORIA CIME
0031' CD 00B0'	CALL WBYTE ;	AKTUALIS BYTE FELIRASA
0034' 1B	DEC DE ;	BYTEDK SZAMA - 1
0035' 10 FB	DJNZ WRREC3	=NINCS TELE A BLOKK
0037' 79	LD A,C ;	(A)=CHECKSUM
0038' CD 00B4'	CALL OBYTE ;	CHECKSUM FELIRASA
003B' 7A	LD A,D ;	ELOKESZIT. VIZSGALATRA
003C' B3	OR E ;	(D) & (E) = 0 ?
003D' C1	POP BC ;	REG. VISSZAMENTES
003E' 20 C5	JR NZ,WRREC1 ;	=VAN MEG KIVIENDO BYTE
0040' C9	RET	

WRREC3:

; WRHEAD - HEADER-BLOKK FELIRAS

; REGISZTEREK:

; BE: (HL)=FILE-NEV LEIRO CIM
 ; (DCB+26)=A FILE TIPUSA
 ; KI: (DCB+28)=1
 ; (HL),(DE),(BC),(AF)=?
 ; (DCB+26)=*
 ; STK: 10 BYTE

0041' CD 00E4'	WRHEAD:	CALL MTRON ;	MOTOR BEKAPCSOLAS
0044' 06 05		LD B,5 ;	4 SEC. CSEND FELIRASA
0046' C5	SAVE1:	PUSH BC ;	IDOKONSTANS MENTESE
0047' 01 0000		LD BC,0 ;	BELSO IDOZITO CIKLUS
004A' 0B	SAVE2:	DEC BC ;	KONSTANS CSOKKENTESE
004B' 7B		LD A,B ;	ELOKESZIT. VIZSGALATRA
004C' B1		OR C ;	(B) & (C) = 0 ?
004D' 20 FB		JR NZ,SAVE2 ;	=VAN MEG IDO !
004F' C1		POP BC ;	IDOKONST. VISSZATOLTES
0050' 10 F4		DJNZ SAVE1 ;	=VAN MEG IDO !
0052' 01 0200		LD BC,512 ;	FILE SZINKRON FELIRASA
0055' 3E AA	SAVE3:	LD A,0AAH ;	SZINKRON BYTE = 0AAH
0057' CD 00B4'		CALL OBYTE ;	1 DB SZINKRON BYTE KI
005A' 0B		DEC BC ;	DARABSZAM CSOKKENTESE
005B' 7B		LD A,B ;	ELOKESZIT. VIZSGALATRA
005C' B1		OR C ;	(B) & (C) = 0 ?

----- FOGGELÉK -----

005D' 20 F6	JR NZ,SAVE3	;=VAN MEG BYTE !
005F' CD 009D'	CALL WRSYN	;BLOKK SZINKRON FELIRAS
0062' 3A 001A*	LD A,(DCB+26)	; (A)=FILE TIPUSA
0065' CD 00B4'	CALL OBYTE	;FILE TIPUS FELIRASA
0068' AF	XOR A	;REKORD SORSZAM=0
0069' 4F	LD C,A	;CHKSUM=0
006A' 32 001C*	LD (DCB+28),A	;REKORD SORSZAM MENTESE
006D' CD 00B4'	CALL OBYTE	;REKORD SORSZAM FELIRAS
0070' 7E	LD A,(HL)	; (A)=FILE NEV HOSSZ
0071' CD 00B0'	CALL WBYTE	;FILE NEV HOSSZ FELIRAS
0074' 47	LD B,A	; (B)=KARAKTEREK SZAMA
0075' 23	INC HL	; (HL)=FILE NEV CIME
0076' 7E	LD A,(HL)	;CIM ALSO BYTE BE
0077' 23	INC HL	;FELSO BYTE CIME
0078' 66	LD H,(HL)	;CIM FELSO BYTE BE
0079' 6F	LD L,A	; (HL)=NEV 1. KAR. CIME
007A' 7E	LD A,(HL)	;FILE NEV FELIRASA
007B' CD 00B0'	CALL WBYTE	;AKTUALIS KAR. FELIRASA
007E' 23	INC HL	; (HL)=KOV. KAR. CIME
007F' 10 F9	DJNZ SAVE4	;=VAN MEG KARAKTER !
0081' 79	LD A,C	; (A)=HEADER CHECKSUM
0082' C3 00B4'	JP OBYTE	;CHECKSUM FELIRASA

SAVE4:

; ENDREC - END-BLOKK FELIRASA

; REGISZTER:

; BE: (DCB+26)=A BLOKK TIPUSA
 ; (DCB+28)=A BLOKK SORSZAMA
 ; KI: (DCB+28)=BLOKK SORSZAM+1
 ; (DCB+26)=*
 ; (HL),(DE),(BC),(AF)=?

; STK: 10 BYTE

; ENDREC:

0085' CD 009D'	CALL WRSYN	;END-BLOKK SZINKRON KI
0088' 3A 001A*	LD A,(DCB+26)	; (A)=BLOKK TIPUS
008B' CD 00B4'	CALL OBYTE	;BLOKK TIPUS FELIRASA
008E' 3A 001C*	LD A,(DCB+28)	; (A)=REGI BLOKK SORSZAM
0091' C6 01	ADD A,1	; (A)=UJ BLOKK SORSZAM
0093' 27	DAA	;A BLOKK SORSZAM BCD !
0094' CD 00B4'	CALL OBYTE	;BLOKK SORSZAM FELIRASA
0097' CD 00B4'	CALL OBYTE	;CHECKSUM FELIRASA
009A' C3 00E6'	JP MTRFF	;MOTOR KIKAPCSOLASA

; WRSYN - BLOKK SZINKRON FELIRASA

; REGISZTEREK:

; BE: NINCS
 ; KI: (HL),(DE),(C)=*
 ; (B)=0
 ; (A)=0D3H

----- FOGSELEK -----

```

; STK:      8 BYTE
;
009D' 06 60   WRSYN:  LD  B,96      ;BLOKK SZINKRON=96*OFFH
009F' 3E FF   LD  A,OFFH      ;(A)=SZINKRON BYTE
00A1' CD 00B4' SAVE6:  CALL OBYTE   ;SZINKRON BYTE FELIRASA
00A4' 10 FB   DJNZ SAVE6      ;=VAN MEG BYTE !
00A6' 3E D3   LD  A,0D3H      ;SZINKRONIZALO KAR.=D3H
00A8' 06 03   LD  B,3        ;IGY BIZTONSAGOSABB !
00AA' CD 00B4' SAVE7:  CALL OBYTE   ;SZINKR. KAR. FELIRASA
00AD' 10 FB   DJNZ SAVE7      ;=VAN MEG SZINKR. KAR.!
00AF' C9      RET

;
; WBYTE - EGY BYTE FELIRASA
;
; REGISZTEREK:
; BE:      (A)=FELIRANDO BYTE
;          (C)=EDDIGI CHECKSUM
; KI:      (HL),(DE),(B),(AF)=*
;          (C)=UJ CHECKSUM
; STK:      6 BYTE
;
00B0' F5      WBYTE:  PUSH AF      ;REGISZTER MENTESE
00B1' 81      ADD  A,C          ;(C)=REGI CHECKSUM
00B2' 4F      LD  C,A          ;(C)=UJ CHECKSUM
00B3' F1      POP  AF          ;REG. VISSZAMENTES
00B4' C5      OBYTE:  PUSH BC      ;REGISZTER MENTESE
00B5' 0E 08   LD  C,8          ;1 BYTE=8 BIT
00B7' 07      OBIT:  RLCA        ;CY<-BIT
00B8' F5      PUSH AF      ;FELIRANDO BYTE MENTESE
00B9' 06 2E   LD  B,46         ;312 MIKROSEC. IDOZITES
00BB' 38 02   JR  C,OBIT1      ;=BIT=1 !
00BD' 06 8A   LD  B,138        ;938 MIKROSEC. IDOZITES
00BF' C5      OBIT1:  PUSH BC      ;IDOKONSTANS MENTESE
00C0' 3A 0000* LD  A,(PORT)     ;(A)=I/O PORT ALLAPOTA
00C3' F6 03   OR  3            ;(PORT)=MAGAS SZINT
00C5' DD CB 14 5E BIT 3,(IX+20) ;HANGADAS ENGEDELVEZE?
00C9' 20 02   JR  NZ,OBIT11    ;=HANG LETILTVA !
00CB' F6 10   OR  10H          ;HANG ENGEDELVEZE !
00CD' D3 1E   OBIT11: OUT (1EH),A ;SZINT KI A VONALRA
00CF' 00      OBIT2:  NOP        ;IDOZITES
00D0' 10 FD   DJNZ OBIT2      ;=VAN MEG IDO !
00D2' E6 EC   AND  0ECH        ;(PORT)=ALACSONY SZINT
00D4' D3 1E   OUT (1EH),A      ;SZINT KI A VONALRA
00D6' C1      POP  BC          ;IDOKONST. VISSZAMENTESE
00D7' 00      OBIT3:  NOP        ;IDOZITES
00D8' 10 FD   DJNZ OBIT3      ;=VAN MEG IDO !
00DA' F6 01   OR  01H          ;(PORT)=NULLPOTENCIAL
00DC' D3 1E   OUT (1EH),A      ;SZINT KI A VONALRA
00DE' F1      POP  AF          ;(A)=FELIRANDO BYTE
00DF' 0D      DEC  C           ;BITEK SZAMANAK CSOKK.

```

----- FOGGELÉK -----

```

00E0' 20 D5      JR    NZ,OBIT      ;=VAN MEG BIT
00E2' C1         POP    BC          ;REG. VISSZAMENTESE
00E3' C9         RET

;
; MTRON - MAGNETOFON MOTOR BEKAPCSOLASA
;
00E4' AF      MTRON:  XOR    A          ;FLAG<-MTRON
00E5' 3E      DEF    B 3EH          ;LD A,-

;
; MTROFF - MAGNETOFON MOTOR KIKAPCSOLASA
;
00E6' 37      MTROFF: SCF          ;FLAG<-MTROFF
00E7' 3A 0000* LD    A,(PORT)      ;(A)=I/O PORT ALLAPOTA
00EA' CB 97    RES    2,A          ;TAVVEZ<-BE
00EC' 30 02    JR     NC,$+4        ;=MTRON FOLYIK
00EE' F6 04    OR     4            ;TAVVEZ<-KI
00F0' D3 1E    OUT    (1EH),A      ;SZINT KI AVONALRA
00F2' 32 0000* LD    (PORT),A      ;STATUSZ MENTESE
00F5' C9      RET

;
; RDHEAD - HEADER KERESÉS/BEOLVASÁS
;
; REGISZTEREK:
; BE:      (HL)=FILE-NEV LEIRO CIME
;          (DCB+20)=MAGNO ALLAPOTJELZO BYTE
;          (DCB+26)=BLOKK TIPUSA
; KI:      (DCB+27)=HIBAS BLOKKOK SZAMA
;          (DCB+29)=FAZISFORDITAS FLAG
;          (DCB+30)=SZAMITOTT BIT IDO
;          (HL),(DE),(BC),(AF)=?
; STK:     24 BYTE
;
00F6' 3E 06      RDHEAD: LD    A,6          ;(A)=DISP. VEZERLO BYTE
00F8' CD 029F'   CALL    CASDSP        ;PRECLEAR BEKAPCSOLASA
00FB' CD 00E4'   CALL    MTRON        ;MOTOR BEKAPCSOLAS
00FE' E5         PUSH    HL            ;FILE NEVLEIRO CIM MENT
00FF' AF      LOAD0:  XOR    A
0100' 32 001B*   LD    (DCB+27),A      ;HIBAS BLOKKOK SZAMA=0
0103' 11 0F00   LD    DE,0F00H        ;KURZOR POZICIO=15.SOR
0106' CD 02A7'   CALL    CASDIR        ;LEGELSO KARAKTERE
0109' CD 0279'   CALL    SPAUSE        ;FILE HEADER KERESESE
010C' 0E 04     LD    C,4            ;MAGNO AZONOSITAS KEZD.
010E' 1E 00     LD    E,0
0110' 53      IDEN11: LD    D,E          ;FAZISFORDITAS FLAG<-0
0111' DB 1F     IDENT1: IN    A,(1FH)    ;MAGAS SZINT KIVARASA
0113' A1         AND    C
0114' 2B FB     JR     Z,IDENT1        ;=MEG ALACSONY SZINT
0116' 43      IDEN10: LD    B,E          ;IDOSZAMLALO<-0
0117' DB 1F     IDENT2: IN    A,(1FH)    ;ETALON MAGAS MERESE
0119' 04         INC    B              ;IDOSZAMLALO NOVELESE

```


----- FOGGELÉK -----

011A' AB		XOR E	
011B' A1		AND C	; MAGNETOFON VONAL VIZSG.
011C' C2 0117'		JP NZ, IDENT2	; =MEG MAGAS SZINT
011F' 78		LD A, B	; (A)=IDOSZAMLALO
0120' FE 2E		CP TETA2	; MAGAS SZINT VIZSGALATA
0122' 38 ED		JR C, IDENT1	; =MAGAS SZINT(2*IDOALAP
0124' FE 8A		CP TETA6	
0126' 30 EB		JR NC, IDEN11	; =MAGAS SZINT)=6*IDOALAP
0128' 43		LD B, E	; IDOSZAMLALO(-O
0129' DB 1F	IDENT3:	IN A, (1FH)	; ETALON ALACSONY MERESE
012B' 04		INC B	; IDOSZAMLALO NOVELESE
012C' AB		XOR E	
012D' A1		AND C	; MAGNETOFON VONAL VIZSG.
012E' CA 0129'		JP Z, IDENT3	; =MEG ALACSONY SZINT
0131' 78		LD A, B	; (A)=IDOSZAMLALO
0132' 15		DEC D	; PULZUSSZAMLALO-1
0133' FE 21		CP TETA15	
0135' 38 06		JR C, IDENT4	; =AKTIV MAGAS SZINT
0137' 14		INC D	; PULZUSSZAMLALO+1
0138' FE 8A		CP TETA6	; ZAVARJEL VIZSGALATA
013A' 30 D4		JR NC, IDEN11	; =TUL HOSSZU ALACSONY
013C' 14		INC D	; PULZUSSZAMLALO+1
013D' 7A	IDENT4:	LD A, D	; (A)=PULZUSSZAM
013E' FE 10		CP 16	; FORDIT FAZIST?
0140' 28 04		JR Z, IDENT8	; =NEM FORDIT
0142' FE EF		CP 239	
0144' 20 D0		JR NZ, IDEN10	; =TUL KEVES MINTA !
0146' 32 001D*	IDENT8:	LD (DCB+29), A	; ALLAPOTJELZO BYTE MENT
0149' 43		LD B, E	; CIKLUSSZAM=256
014A' 6B		LD L, E	; ALAPIDOSSZEG TAROLO
014B' 63		LD H, E	; TAROLO TORLESE
014C' D5	IDENT7:	PUSH DE	; ALAPIDO MERESE
014D' DB 1F	IDENT5:	IN A, (1FH)	; MAGAS SZINT KIVARASA
014F' AA		XOR D	; FAZISFORD. KORREKCIO
0150' A1		AND C	; MAGNETOFON VONAL VIZSG.
0151' CA 014D'		JP Z, IDENT5	; =MEG ALLACSONY SZINT
0154' 1E 00		LD E, 0	; IDOSZAMLALO(-O
0156' DB 1F	IDENT6:	IN A, (1FH)	; MAGASSZINT MERESE
0158' 1C		INC E	; IDOSZAMLALO NOVELESE
0159' AA		XOR D	; FAZISFORD. KORREKCIO
015A' A1		AND C	; MAGNETOFON VONAL VIZSG.
015B' C2 0156'		JP NZ, IDENT6	; =MEG MAGAS SZINT
015E' 7B		LD A, E	; (A)=IDOALAP
015F' FE 8A		CP TETA6	; ZAVARJEL VIZSGALATA
0161' 30 EA		JR NC, IDENT5	; =TUL HOSSZU
0163' 16 00		LD D, 0	; FELSO BYTE (-O
0165' 19		ADD HL, DE	; IDOALAP OSSZEGZESE
0166' D1		POP DE	; (D)=MASZK, (E)=0
0167' 10 E3		DJNZ IDENT7	; =MEG NINCS VEGE
0169' 7C		LD A, H	; (A)=IDOALAP OSSZEG/256

016A' 32 001E*		LD (DCB+30),A ;	MERT IDOALAP MENTESE
016D' CD 0235'		CALL RDSYN ;	SZINKRON KERESSE
0170' DD BE 1A		CP (IX+26) ;	BEOLVASOTT TIPUS VIZSG
0173' 20 8A		JR NZ,LOAD0 ;	=NEM FILENEV, ILLEGALIS
0175' 53		LD D,E ;	CHECKSUM=BLOKK SORSZ.=0
0176' CD 026B'		CALL INBYTE ;	BLOKK SORSZAM BEOLV.
0179' B7		OR A ;	BLOKK SORSZAM=0 ?
017A' C2 00FF'		JP NZ,LOAD0 ;	=NEM FILE-NEV BLOKK
017D' CD 026B'		CALL INBYTE ;	FILE-NEV HOSSZ BEOLV.
0180' 47		LD B,A ;	(B)=KARAKTEREK SZAMA
0181' 4F		LD C,A ;	KAR. SZAMANAK MENTESE
0182' 21 0000*		LD HL,NAMERM ;	(HL)=FILE-NEV PUFF.CIME
0185' CD 026B'	LOAD1:	CALL INBYTE ;	FILE-NEV BEOLVASASA
0188' 77		LD (HL),A ;	AKTUALIS KAR. MENTESE
0189' 23		INC HL ;	PUFFERCIM+1
018A' 10 F9		DJNZ LOAD1 ;	=MEG VAN A NEVBOL
018C' 3E 10		LD A,10H ;	FILE-NEV HOSSZA MAX. 16
018E' 91		SUB C ;	NEV HOSSZANAK VIZSG.
018F' 28 06		JR Z,LOAD3 ;	=A NEV 16 KAR. HOSSZU
0191' 47		LD B,A ;	(B)=HOSSZ KULONBSEGE
0192' 36 20	LOAD4:	LD (HL),20H ;	A PUFFER HATRALEVO
0194' 23		INC HL ;	RESZET FELT. SPACEVAL
0195' 10 FB		DJNZ LOAD4 ;	=VAN MEG TAR
0197' 11 0F08	LOAD3:	LD DE,0F08H ;	KURZOR POZICIONALASA
019A' CD 02A7'		CALL CASDIR ;	15. SOR , 9. KARAKTER
019D' CD 026B'		CALL INBYTE ;	CHECKSUM BEOLVASASA
01A0' E1		POP HL ;	(HL)=FILE-NEV LEIRO CIM
01A1' DD CB 14 56		BIT 2,(IX+20) ;	VOLT FILE-NEV MEGADVA ?
01A5' 28 17		JR Z,FOUND ;	=NEM VOLT MEGADVA
01A7' E5		PUSH HL ;	PUFFER KEZDOCIM MENT.
01AB' 11 0000*		LD DE,NAMERM ;	(DE)=FILE-NEV PUFF. CIM
01AB' 46		LD B,(HL) ;	(B)=FILE-NEV HOSSZA
01AC' 79		LD A,C ;	(A)=MAGN. BEOLV. HOSSZ
01AD' B8		CP B ;	FILE-NEV HOSSZ VIZSG.
01AE' 38 1F		JR C,SKIP ;	=MAGNOROL ROVIDEBB
01B0' 23		INC HL	
01B1' 7E		LD A,(HL) ;	FILE-NEV CIM ALSO BYTE
01B2' 23		INC HL	
01B3' 66		LD H,(HL) ;	FILE-NEV CIM FELSO BYTE
01B4' 6F		LD L,A	
01B5' 1A	LOAD2:	LD A,(DE) ;	FILE-NEVEK HASONLITASA
01B6' BE		CP (HL)	
01B7' 20 16		JR NZ,SKIP ;	=NEM EGYFORMA NEVEK
01B9' 13		INC DE	
01BA' 23		INC HL	
01BB' 10 FB		DJNZ LOAD2 ;	=VAN MEG A NEVEKBOL
01BD' E1		POP HL	
01BE' 21 0292'	FOUND:	LD HL,TFOUND ;	MEGVAN A FILE !
01C1' 06 07		LD B,LFOUND ;	(B)=SZOVEG HOSSZA
01C3' CD 028A'		CALL WRMES ;	SZOVEG A DISPLAYRE

----- FOGGELÉK -----

```

01C6' 21 0000*      LD  HL,NAMERM  ; (HL)=NEVPUFFER CIME
01C9' 06 10          LD  B,16      ; (B)=FILENEV HOSSZ
01CB' CD 028A'       CALL WRMES    ; SZOVEG A DISPLAYRE
01CE' C9             RET
01CF' 21 0299'       SKIP:        LD  HL,TSKIP  ; NEM A KERESETT FILE !
01D2' 06 06          LD  B,LSKIP   ; (B)=SZOVEG HOSSZA
01D4' CD 028A'       CALL WRMES    ; SZOVEG A DISPLAYRE
01D7' 21 0000*      LD  HL,NAMERM  ; (HL)=NEVPUFFER CIME
01DA' 06 10          LD  B,16      ; (B)=FILENEV HOSSZ
01DC' CD 028A'       CALL WRMES    ; SZOVEG A DISPLAYRE
01DF' C3 00FF'       JP   LOAD0    ; KOV. FILE KERESESE

;
; CHKSUM - CHECKSUM ELLENORZES
;
; REGISZTEREK:
; BE:      (DCB+27)=EDDIGI HIBAS BLOKK SZAM
; KI:      (HL),(DE),(BC),(AF)=?
; STK:     24 BYTE
;
01E2' D5             CHKSUM:      PUSH DE      ; REGISZTER MENTESE
01E3' CD 026B'       CALL INBYTE  ; CHECKSUM BEOLVASASA
01E6' D1             HELP1:      POP  DE      ; (D)=KEPZETT CHECKSUM
01E7' BA             CP  D        ; KEPZETT=OLVASOTT?
01E8' 3A 001B*       LD  A,(DCB+27) ; (A)=HIBA SZAMLALO
01EB' 28 06          JR  Z,CHKO    ; =NINCS HIBA
01ED' C6 01          ADD  A,1      ; HIBA !, HIBA SZAMLALO+1
01EF' 27             DAA          ; HIBA SZAMLALO BCD !
01F0' 32 001B*       LD  (DCB+27),A ; HIBA SZAMLALO MENTESE
01F3' 57             CHK0:       LD  D,A    ; REKSZAM,CHKSUM KIIRASA
01F4' 3E 0E          LD  A,0EH     ; KURZOR A SOR ELEJERE
01F6' CD 029F'       CALL CASDSP
01F9' 7B             LD  A,E      ; BLOKK SORSZAM KIIRASA
01FA' CD 020D'       CALL NUMBHI   ; FELSO SZAMJEGY
01FD' 7B             LD  A,E
01FE' CD 0211'       CALL NUMBLO   ; ALSO SZAMJEGY
0201' 3E 20          LD  A,' '
0203' CD 029F'       CALL CASDSP    ; SPACE KIIRASA
0206' 7A             LD  A,D      ; HIBA SZAMLALO KIIRASA
0207' CD 020D'       CALL NUMBHI   ; FELSO SZAMJEGY
020A' 7A             LD  A,D
020B' 18 04          JR  NUMBLO    ; ALSO SZAMJEGY + RETURN

;
020D' 0F             NUMBHI:      RRCA      ; FELSO SZAMJEGY KIIRASA
020E' 0F             RRCA      ; FELSO SZAMJEGY LETO-
020F' 0F             RRCA      ; LASA AZ ALSO HELYERE
0210' 0F             RRCA

;
0211' E6 0F          NUMBLO:      AND  0FH   ; ALSO SZAMJEGY KIIRASA
0213' F6 30          OR   '0'      ; (A)=ASCII KARAKTER KOD
0215' C3 029F'       JP   CASDSP   ; KARAKTER KIIRASA + RET

```

```

;
; PRGREC - PROGRAM BLOKK BEOLVASASA
;
; REGISZTEREK:
; BE:      (DCB+20)=ALLAPOTBYTE
;          (BC)=ELTOLASI CIM
; KI:      (BC)=*
;          (HL),(DE),(AF)=?
; STK:     24 BYTE
;
0218' CD 026B' PRGREC: CALL INBYTE ;PROGRAM BLOKK BEOLVASAS
021B' 6F LD L,A ;TOLTESI CIM ALSO BYTE
021C' CD 026B' CALL INBYTE ;TOLTESI CIM FELSO BYTE
021F' 67 LD H,A ;(HL)=TOLTESI CIM
0220' 09 ADD HL,BC ;TOLTESI CIM + OFFSZET
0221' CD 026B' CALL INBYTE ;BLOKK HOSSZ BEOLVASASA
0224' 47 LD B,A ;(B)=BYTEOK SZAMA
0225' CD 026B' PRG1: CALL INBYTE ;ADAT BYTE BEOLVASASA
0228' DD CB 14 66 BIT 4,(IX+20) ;LOAD/TEST FUNKCIO ?
022C' 20 01 JR NZ,PRG2 ;=TEST FUNKCIO
022E' 77 LD (HL),A ;ADAT BYTE A MEMORIABA
022F' 23 PRG2: INC HL ;TOLTESI CIM NOVELESE
0230' 10 F3 DJNZ PRG1 ;=VAN MEG BYTE !
0232' C3 01E2' JP CHKSUM ;CHECKSUM ELLENORZ.+RET
;
; RDSYN - SZINKRON BYTE KERESSES
;
; REGISZTEREK:
; BE:      NINCS
; KI:      (HL),(DE),(BC)=*
;          (A)=KOVETKEZO BLOKK TIPUSA
; STK:     12 BYTE
;
0235' AF RDSYN: XOR A ;(A)<-0 , (CY)<-0
0236' CD 0245' RDS1: CALL INBIT ;KOVETKEZO BIT BEOLVASAS
0239' FE D3 CP OD3H ;SZINKRON KARAKTER ?
023B' 20 F9 JR NZ,RDS1 ;=MEG NEM SZINKRON !
023D' CD 026B' RDS2: CALL INBYTE ;KOV. BYTE BEOLVASASA
0240' FE D3 CP OD3H ;TOVABBI SZINKRON KAR.
0242' 28 F9 JR Z,RDS2 ;KISZURESE,TIPUS BEOLV.
0244' C9 RET
;
; INBIT - EGY BIT BEOLVASASA
;
; REGISZTEREK:
; BE:      (DCB+29)=FAZISFORDITAS FLAG
;          (DCB+30)=SZAMITOTT BIT IDO
; KI:      (HL),(DE),(BC),(A)=*
;          (CY)=BEOLVASOTT BIT
; STK:     6 BYTE

```

----- FOGGELÉK -----

```

;
0245' C5      INBIT:    PUSH BC      ;REGISZTEREK MENTESE
0246' D5      PUSH DE
0247' F5      PUSH AF
0248' ED 5B 001D* LD DE,(DCB+29);(D)=IDO,(E)=FAZISFORD.
024C' 01 0004    INB2:    LD BC,4      ;(B)=IDOSZLO,(C)=MASZK
024F' DB 1F      INB3:    IN A,(1FH)    ;MAGAS SZINT KIVARASA
0251' AB      XOR E      ;FAZISFORD. KORREKCIO
0252' A1      AND C      ;MAGNETOFON VONAL VIZSG.
0253' 28 FA      JR Z,INB3    ;=MEG ALACSONY SZINT
0255' DB 1F      INB4:    IN A,(1FH)    ;MAGAS SZINT MERESE
0257' 04      INC B      ;IDO SZAMLALO NOVELESE
0258' AB      XOR E      ;FAZISFORD. KORREKCIO
0259' A1      AND C      ;MAGNETOFON VONAL VIZSG.
025A' 20 F9      JR NZ,INB4    ;=MEG MAGAS SZINT
025C' 78      LD A,B      ;(A)=MERT BIT IDO
025D' FE 06      CP 6      ;ZAJ KISZURESE
025F' 38 EB      JR C,INB2    ;=ZAJ VOLT !
0261' BA      CP D      ;BIT IDO VIZSGALAT
0262' CB 10      RL B      ;(CY)=BEOLVASOTT BIT
0264' F1      POP AF      ;(A) VISSZAMENTESE
0265' CB 18      RR B
0267' 17      RLA      ;BEOLV. BIT AZ ACO-BAN
0268' D1      POP DE      ;REGISZTEREK VISSZAMENT.
0269' C1      POP BC
026A' C9      RET

```

```

;
; INBYTE - EGY BYTE BEOLVASASA
;

```

```

; REGISZTEREK:
; BE:      (D)=CHECKSUM
; KI:      (A)=BEOLVASOTT BYTE
;          (D)=UJ CHECKSUM
;          (HL),(E),(BC)=*
; STK:     10 BYTE
;

```

```

026B' C5      INBYTE:    PUSH BC      ;REGISZTER MENTESE
026C' 06 08      LD B,8      ;1 BYTE = 8 BIT
026E' CD 0245'    INB1:    CALL INBIT    ;KOVETKEZO BIT BEOLV.
0271' 10 FB      DJNZ INB1    ;=VAN MEG BIT !
0273' 47      LD B,A      ;BEOLVASOTT BYTE MENTESE
0274' 82      ADD A,D      ;UJ CHECKSUM KEPZESE
0275' 57      LD D,A      ;(D)=CHECKSUM
0276' 78      LD A,B      ;(A)=BEOLVASOTT BYTE
0277' C1      POP BC      ;REGISZTER VISSZAMENT.
0278' C9      RET

```

```

;
; SPAUSE - SZUNET KERESÉS A SZALAGON
;

```

```

0279' C5      SPAUSE:    PUSH BC      ;REGISZTER MENTESE

```

----- FOGGELÉK -----

027A'	01 00E0	SPA0:	LD	BC,0EOH	; (BC)=IDOKONSTANS
027D'	DB 1F	SPA1:	IN	A,(1FH)	; MAGAS SZINT KIVARASA
027F'	E6 04		AND	4	; MAGNETOFON VONAL VIZSG.
0281'	28 F7		JR	Z,SPA0	; =MEG ALACSONY SZINT
0283'	0B		DEC	BC	; IDO SZAMLALO CSOKKENT.
0284'	78		LD	A,B	; ELOKESZITES VIZSGALATRA
0285'	B1		OR	C	; (B) & (C) = 0 ?
0286'	20 F5		JR	NZ,SPA1	; =VAN MEG IDO !
0288'	C1		POP	BC	; REGISZTER VISSZAMENT.
0289'	C9		RET		

;

; WRMES - SZOVEG A DSP UTOLSO SORABA

028A'	7E	WRMES:	LD	A,(HL)	; (A)=AKT. KARAKTER
028B'	CD 029F'		CALL	CASDSP	; KARAKTER KIIRATASA
028E'	23		INC	HL	; KOVETKEZO KAR. CIME
028F'	10 F9		DJNZ	WRMES	; =VAN MEG KARAKTER
0291'	C9		RET		

0292'	46 4F 55 4E	TFOUND:	DEFM	'FOUND: '
0296'	44 3A 20			
0007		LFOUND	EQU	7

0299'	53 4B 49 50	TSKIP:	DEFM	'SKIP: '
029D'	3A 20			
0006		LSKIP	EQU	6

;

; CASDSP - FELTETELES KIIRAS A DSP-RE

029F'	DD CB 14 4E	CASDSP:	BIT	1,(IX+20)	; FELTETEL VIZSGALAT
02A3'	CA 0000*		JP	Z,DISP	; =KIIRATAS ENGEDELYEZVE
02A6'	C9		RET		

;

; CASDIR - FELTETELES KURZOR CIMZES

02A7'	DD CB 14 4E	CASDIR:	BIT	1,(IX+20)	; FELTETEL VIZSGALAT
02AB'	CA 0000*		JP	Z,DIRCUR	; =KIIRATAS ENGEDELYEZVE
02AE'	C9		RET		

INDEX

! I-3, I-5, I-37
 " I-3
 ‡ I-3, I-4, I-5, I-37
 § I-3, I-5, I-36
 §§ I-39
 ζ I-3, I-5, I-40
 ζ...ζ I-37
 & I-3, F-36
 ' I-3, I-42
 (..... I-3, I-6, I-8, I-13
) I-3, I-6, I-8, I-13
 * I-3, I-8, I-13, II-1
 * (AUTO parancs esetén) I-19
 ** I-39
 **\$ I-39
 + I-3, I-8, I-11, I-13, I-38, II-1
 , I-3, I-35, I-39
 - I-3, I-8, I-13, I-38, II-1
 I-2, I-3, I-4, I-14, I-38
 I-19
 / I-3, I-8, I-13, II-1
 : I-2, I-3, I-14
 ; I-3, I-30, I-35
 < I-3, I-9, I-12, I-19
 = I-3, I-9, I-12, I-31, II-1
 > I-3, I-9, I-12, I-19
 ? I-3, I-30, I-36
 ?? I-30
 [..... I-19
] I-19
 † I-3, I-8, I-13, II-1
 †††† I-39
 ‡ I-14, II-54, F-10
 SHIFT/‡ I-15, II-54, F-13
 → I-14
 SHIFT/→ I-15, II-54, F-10, F-13
 ↓ I-14, I-16, II-51, F-10
 | I-19

A

ABS(x) függvény I-47, II-46
 Adatállomány kezelés I-24, I-31
 I-34, I-41, I-44, I-45, II-55
 - név ... I-24, I-34, I-44, I-46, II-55
 - záró-blokk II-58
 Adat-blokk II-55, II-57

- beolvasás - PRGREG ... II-65, F-55
 F-94
 - felírás - WRREC . II-63, F-55, F-86
 Aktuális sor I-2, I-14
 Aláhúzás bekapcsolás - CTR/E .. F-10
 - kikapcsolás - CTR/U F-12
 Alsó index bekapcsolás - CTR/P F-11
 - kikapcsolás - CTR/Q F-12
 AND művelet I-10, I-13, II-22
 Aritmetikai munkaregiszterek . II-36
 - műveletek I-8
 - operátor csomag II-23
 - rutinok II-39
 - egész műveletek II-40
 - egyszeresponos műveletek II-41
 - duplapontos műveletek ... II-43
 ASC(x\$) függvény I-47, F-2
 ASCII kódok F-7
 ATN(x) függvény I-47, II-44
 ATN - soros busz figyelemfelhívás
 F-41, F-44
 AUTO parancs I-19, F-2
 Automatikus indítású gépi kódú prog-
 ramállomány záró blokk II-58
 - sorszámzás I-19
 - típuskonverzió ... I-7, I-47, II-21

A

Allomány kezelés I-24, I-31
 I-34, I-41, I-44, I-45, II-55
 - leírás (FD) hiba I-24, I-31
 I-41, F-5, F-6
 - név I-24, I-32, I-34, I-44
 I-46, II-57
 - szinkron mező II-56, II-61

B

BASIC interpreter II-1, II-4
 - program tárolás I-44, II-1
 II-3
 - változók I-5, II-6
 - végrehajtás I-21, II-4
 BEEP utasítás I-22, F-23
 - elemi hanggenerátor F-79

BE-1 Bemeneti kapu F-42
 BE-2 Bemeneti kapu F-43
 Bemeneti/Kimeneti kapuk F-39
 Betelt a karakterterület (OS) hiba
 I-23,F-4,F-6
 Betelt a memória (OM) hiba I-14
 I-26,F-3,F-6
 Betöltési cím II-58
 Betűk I-2
 BIE csatlakozó F-42,F-44
 Billentyűk I-14,I-16
 - periféria címei F-15
 Billentyűzet bemenet F-43
 - kezelő rutinok - KYBHND, KKBHND
 II-51,II-54,F-54,F-68,F-71
 - különleges mód ROM-kapocs . II-51
 F-30
 - puffer II-4,II-5,F-28
 Bináris - karakteres konverzió II-38
 Bit magnóra írása II-59
 - olvasása II-60
 Blokk II-55
 - adat-blokk II-57
 - név-blokk II-57
 - záró-blokk II-58
 - sorszám II-56
 - szabványos felépítése II-56
 - szinkronizálás - RDSYN II-65
 F-94
 - szinkron mező II-56,II-62
 - típus II-56,II-57
 Bonyolult karakteres kifejezés (ST)
 hiba I-47,F-4,F-6
 Botkormány F-39,F-43,F-44
 Break in nnnn üzenet I-17,I-26
 I-45
 BRK billentyű I-15,I-16,I-20
 I-21,I-30,II-4,II-54
 BS-Bad Subscript hiba I-26,F-3
 F-6
 Byte beolvasás - INBYTE II-65
 F-55,F-95

C

CALL(i[,j]) függvény I-47
 CDBL(x) függvény I-48,II-43
 CHR\$(1) vezérlő kód F-9

CHR\$(9) vezérlő kód F-10
 CHR\$(i) függvény I-48,F-2
 Cimfüggvény II-11
 CINT(x) függvény I-49,II-41
 CLEAR utasítás I-23,I-33,II-14
 F-2,F-4
 CLOSE utasítás I-24,F-5
 CLRDOT - grafikus pont kikapcsolása
 II-67,F-55,F-83
 CLS billentyű I-16,II-54,F-11
 CLS utasítás I-24
 CN-Can't Continue hiba I-20,F-4
 F-6
 Commodore soros busz vezérlés . F-41
 F-43
 CONT parancs I-20,I-31,I-45
 II-6,F-4,F-5
 COS(x) függvény I-49,II-44
 CREATE utasítás I-24,I-41
 F-2,F-5
 CSNG(x) függvény I-50
 Csatlakozók F-39
 - BIE F-42,F-44
 - botkormány F-39,F-43,F-44
 - soros busz F-41,F-43,F-44
 - TAPE F-39,F-40,F-42
 CTR billentyű I-16
 - /B nyújtott karakter megjelenítés
 bekapcsolás F-9
 - /C fehér képernyő alapszín .. F-9
 - /D inverz karakteralap bekapcsol-
 lás F-9
 - /E karakter aláhúzás bekapcsolás
 F-10
 - /F előtörítés bekapcsolás ... I-24
 F-10
 - /G hangjelzés F-10
 - /N kurzor a sor elejére F-11
 - /O függőleges írás F-11
 - /P alsó index F-11
 - /Q felső index F-12
 - /R normál méretű karakter megje-
 lenítés F-12
 - /S fekete képernyő alapszín beál-
 litás F-12
 - /T normál karakteralap beállítás
 F-12
 - /U karakter aláhúzás kikapcsolás
 F-12
 - /V előtörítés kikapcsolás ... F-12

----- INDEX -----

- /W vízszintes írás F-13

D

Dallamok megszólaltatása F-23
 DATA-lista I-42, I-43
 DATA utasítás I-24, I-42, I-43
 DD-Double Dimensioned hiba I-26
 F-3, F-6
 Decimális pont I-3, I-4, I-38
 DEFDBL utasítás I-25
 DEFINT utasítás I-25
 DEFSNG utasítás I-25
 DEFSTR utasítás I-25
 DELETE utasítás I-20, F-2
 DIM utasítás I-26, II-10
 DIRCUR - direkt kurzor címző rutin
 II-65, F-55, F-81
 Direkt (parancs) mód I-1, I-19
 I-22, I-31, II-4
 Direkt módban nem alkalmazható (ID)
 hiba I-31, F-4, F-6
 /O-Division by Zero hiba I-9, F-4, F-6
 Duplapontos valós konstans I-4
 - műveletek II-42
 - számbábrázolás II-33
 - tábla elem II-9
 DSPHND - Képernyő kezelő rutin
 II-49, II-54, F-54, F-59

E

EDIT parancs I-14, I-20, F-3, F-10, F-13
 Editálás (programszerkesztés) . I-13
 Editor I-13, I-20
 - újraindítása I-14, F-10
 Egész szám I-4
 Egész típusú - karakteres konverzió
 II-38
 - műveletek II-39
 - számbábrázolás II-29
 - tábla elem II-8
 Egyszeresponos konstans .. I-4, II-6
 - műveletek II-41
 - számbábrázolás II-32
 - tábla elem II-9

Egy sort beolvasó rutin - GLINE
 II-53, F-54, F-77
 Egy byte beolvasó rutin - INBYTE
 II-65, F-55, F-95
 Egyvonalas oktáv F-25
 Elágazó utasítás I-10
 Elemi hanggenerátor - BEEP F-79
 Ellenőrző összeg II-57
 ELSE kulcsszó I-29
 Előtörlés bekapcsolás (CTR/F) . F-10
 - kikapcsolás (CTR/V) F-12
 END utasítás I-20, I-26, II-6
 ENDREC - Záró-blokk felírás .. II-64
 F-55, F-88
 Error in nwnn üzenet I-17
 ERL függvény I-49
 ERR függvény I-49, F-1
 ERROR utasítás I-26, F-1, F-2, F-5
 EXP(x) függvény I-50, II-45
 Exponens ábrázolás II-33
 Exponenciális alak I-39, II-32
 - kijelzés I-39
 ?Extra ignored üzenet I-30

É

ékezetes betűk I-2, F-7
 érték-rész (karakteres) II-13

F

FC-Illegal Function Call hiba . I-20
 I-24, I-32, I-34, I-36, I-44, I-45
 I-53, I-55, F-2, F-3, F-6
 FD-File Description hiba I-24
 I-31, I-41, F-5, F-6
 Fehér képernyő alapszín beállítás
 (CTR/C) F-9
 Fekete képernyő alapszín beállítás
 (CTR/S) F-12
 Felhasználói (UE) hiba I-27, F-5
 F-6
 Felhasználói karakterek definiálása
 F-17
 Felső index (CTR/Q) F-12
 File kezelés I-24, I-31, I-34

I-41, I-44, I-45, II-55
 FIX(x) függvény I-50, II-46
 Fixpontos szám I-4
 FOR-adatcsomag felépítése II-16
 II-17
 Formai (SN) hiba I-15, I-25, I-28
 I-50, I-52, F-1, F-6
 Formátum vezérlés (USING) I-36
 Form-feed II-52
 FOR...NEXT ciklus I-23, I-27
 FOR nélküli NEXT (NF) hiba I-28
 F-1, F-6
 FOR utasítás I-27
 FOUND: üzenet I-32
 FRE(0), FRE(" ") függvény I-50
 Függvények I-11, I-13, I-46, II-43

G

Gépi kódú rutinok tárolása ... II-24
 Gépi nulla II-34
 GLINE - egy sort beolvasó rutin
 II-53, F-54, F-77
 GOSUB-adatcsomag felépítése .. II-18
 GOSUB nélküli RETURN (RG) hiba
 I-28, F-1, F-6
 GOSUB utasítás I-23, I-28, I-33
 I-42, F-3
 Grafikus pont kezelő rutinok . II-66
 - bekapcsolás II-67, F-55, F-83
 - kikapcsolás II-67, F-55, F-83
 - vizsgálat II-67, F-55, F-84

H

Hanggenerátor - BEEP F-79
 - jelforma I-23, F-23
 - paraméterezés I-23, F-23
 Hangjelzés (CTR/G) F-10
 Hatványozás-csomag II-23
 Hibajelzések I-17, F-1
 Hibakezelés (programozott) I-33
 Hiba nélküli RESUME (RW) hiba . I-44
 F-5, F-6
 Hibás index (BS) hiba . I-26, F-3, F-6
 Hiányzó operandus (MO) hiba I-9

II-20, F-5, F-6
 Hosszú karaktersorozat (LS) hiba
 I-12, I-47, F-4, F-6

I

I regiszter I-48
 IF...GOTO...ELSE utasítás I-29
 IF...THEN...ELSE utasítás . I-29, F-3
 ID-Illegal Direct hiba I-31, F-4
 F-6
 Illegális funkcióhívás (FC) hiba
 I-20, I-24, I-32, I-34, I-36, I-38
 I-43, I-44, I-45, I-48, I-54, I-56
 Indirekt (program) mód I-1, I-19
 Index számítás II-11
 Indexelt változó I-6
 - tábla elem II-10
 INBYTE - Egy byte beolvasás .. II-65
 F-55, F-95
 Indítási cím II-58
 INKEY\$ függvény I-50
 INIT - Power-On RESET rutin ... F-56
 I-O kapuk F-39
 INP(i) függvény I-51, F-2, F-15
 INPUT utasítás I-30, F-4
 INPUT\$ utasítás .. I-31, I-34, F-4, F-5
 Input portok F-42
 INT(x) függvény I-51, II-46
 Interpreter II-1
 - belső kódjai II-1, F-45
 - működése II-1, II-4
 - ROM-kapcsok F-36, F-37
 - változók F-33
 Inverz alap beállítás (CTR/D) .. F-9
 - kikapcsolás (CTR/T) F-12
 IX regiszter I-48, F-30

J

Jelölések I-19, I-46

INDEX

K

Kalkulátor mód I-1
 Karakter I-2
 - aláhúzás bekapcsolás (CTR/E) F-10
 - aláhúzás kikapcsolás (CTR/U) F-12
 - definiálás F-17
 - egymásra írás F-13
 - generátor terület II-50
 - készlet I-2, F-7
 - kódok F-7
 - konstansok ... I-3, I-24, I-25, I-40
 - műveletek I-11, I-12
 - típusú tábla elem II-10
 - törlése I-14, F-10
 - változó terület I-23, II-14
 - változók kezelése II-12
 Karakteres érték I-46
 Karakterisztika ábrázolása ... II-33
 Kettes komplement kódú szábrázolás
 II-29, II-30
 Képernyő kezelő rutin - DSPHND II-49
 II-54, F-54, F-59
 - kioltás jelző bit F-42
 - lapkijelölés F-40
 - mérete I-43
 - RAM kezdőcíme F-30
 - állomány záró-blokk II-58
 - tartalom mentése (SAVE SCREEN)
 I-44
 - törlése (CLS) I-24, F-11
 Kétvonalas oktáv F-25
 Kifejezés I-8, I-25
 - csomagok II-21, II-23, II-24
 - feldolgozás II-19
 Kimeneti kapuk F-39
 - KI-1 (0...63 címek) F-39
 - KI-2 (64...127 címek) F-41
 Kimerült a DATA-lista (OD) hiba
 I-42, F-2, F-6
 Kis oktáv F-24
 Kitevő (exponens) ábrázolása . II-33
 Kiváráshoz billentyűzet vizsgáló rutin - KKBHND II-51, F-54, F-71
 Kivonó rutinok ... II-40, II-41, II-43
 KKBHND - kiváráshoz billentyűzet kezelő rutin II-51, F-54, F-71
 Konkatenáció (összefűzés) I-11
 Konstansok I-3

Kontra oktáv F-24
 Konverzió I-7
 - automatikus I-7
 - bináris - karakteres II-37, II-38
 - függvények I-48, I-49
 - karakteres - bináris II-37, II-38
 - karakteres - duplapontos .. II-38
 - rutinok . II-37, II-41, II-42, II-43
 - táblázatok F-49
 Kulcsszó I-5, I-19, I-22, F-45
 - belső kódok F-45
 KURPOZ - kurzor pozíció megállapító rutin II-66, F-55, F-80
 Kurzor a sor elejére (CTR/N) .. F-11
 - címző rutin - DIRCUR II-65
 F-55, F-81
 Kurzor pozíciót megállapító rutin - KURPOZ II-66, F-55, F-80
 Különleges billentyűk I-14, I-16
 - karakterek I-2
 KYBHND - billentyűzet vizsgáló rutin
 II-54, F-54, F-68

L

Lebegőpontos szám I-4
 LEFT\$(x\$, i) függvény I-51, F-2
 Leíró rész (karakteres) II-12
 LEN(x\$) függvény I-51
 LET utasítás I-31
 Lezáratlan hibakezelés (NR) hiba
 I-33, F-5, F-6
 LIST parancs I-20
 LLIST parancs I-21
 ln(x) függvény (LOG) I-51, II-45, F-3
 LOAD utasítás I-32, I-45, F-2
 LOG(x) függvény I-51, II-45, F-3
 Logikai állománykezelés .. I-24, I-31
 I-34, I-41, I-44, I-45
 - és művelet I-10
 - műveletek I-9
 - NEM művelet I-10
 - operátor csomag II-23
 - VAGY művelet I-10
 LPRINT utasítás I-32, I-54
 LPRINT USING utasítás I-32
 LS-Long String hiba I-12, I-47
 F-4, F-6

Ludolf-féle szám (PI) I-52

M

Magnó adatbemenet F-42
 - adatkimenet F-40
 - logikai kezelés II-55
 - fizikai kezelés II-59
 - bit felírás II-59
 - bit beolvasás II-60
 - távvezérlés F-39, F-40
MAGNO - magnó kezelő rutinok . II-63
 F-86
 Mantissza ábrázolása II-33
 Maximális laphossz II-52
 - sorhossz II-52
 Megjegyzés (REM) utasítás I-42
 Megjeleníthető karakterek .. F-7, F-8
 Megszakítás ... I-16, I-23, I-45, II-48
 Memória felosztás F-27
MIDS(x\$, i[, j]) függvény ... I-51, F-2
MO-Missing Operand hiba .. I-9, II-20
 F-5, F-6
 Munkaváltozók II-36, F-29, F-33
 Működési módok I-1
 Műveleti jelek I-8, I-9, F-45
 - végrehajtási sorrendje I-12

N

Nagy oktáv F-24
 Nemdefiniált sor (UL) hiba I-14
 I-21, I-28, I-29, I-33, I-43, I-44
 F-3, F-6
 Nem folytatható (CN) hiba I-20
 F-4, F-6
NEW parancs I-16, I-21, I-22
 I-32, I-33
NEXT utasítás I-23, I-27
NF-NEXT without FOR hiba I-28
 II-16, F-1, F-6
 Név-blokk II-55, II-57
 - felírás - **WRHEAD** II-63
 F-55, F-87
 - keresés/olvasás - **RDHEAD** .. II-64
 F-55, F-90

NMI engedélyezés F-25, F-39, F-40
NMI-TIMER rutin II-48, F-55
 Normál alak I-4, II-32
 Normál méretű karakter megjelenítés
 (CTR/R) F-12
NOT művelet I-10, I-13, II-22
NR-No RESUME hiba I-33, F-5, F-6
 Nullával osztás (/0) hiba I-9
 F-4, F-6
 Numerikus konstans ... I-3, I-24, I-25

NY

Nyomtató kezelő rutin - **PRTHND**
 II-52, F-54, F-75
 Nyújtott méretű karakter megjeleni-
 tés (CTR/B) F-9, F-12

O

OD-Out of DATA hiba ... I-42, F-2, F-6
Ok felirat I-1, I-13, I-26
 Oktávok F-24, F-25
OM-Out of Memory hiba I-14, I-26
 F-3, F-6
ON ERROR GOTO utasítás ... I-33, I-44
 I-49, F-3
ON x GOSUB utasítás ... I-33, F-2, F-3
ON x GOTO utasítás I-33, F-2, F-3
OPEN utasítás I-34, F-2, F-6
OR művelet I-10, I-13, II-22
OS-Out of String Space hiba ... I-23
 F-4, F-6
 Osztó rutinok II-40, II-42, II-43
OUT utasítás I-34, F-2
 Output kapuk F-39
OV-Overflow hiba . I-7, I-9, I-10, I-22
 I-34, I-46, I-49, I-50, F-3, F-6

O

Osszeadó rutinok . II-40, II-41, II-43
 Osszefűzés (konkatenáció) I-11
 Osszehasonlító műveletek .. I-9, I-12

----- INDEX -----

- rutinok II-40,II-42,II-43

P

Parancsok I-19
 - végrehajtása II-5
 Parancs mód I-1,I-19,I-22,I-31
 PEEK(x) függvény I-52
 Periféria kezelő rutinok II-47
 PI függvény I-52
 POINT(i,j) függvény I-52,F-2
 POKE utasítás I-34,I-52,F-2
 PONT - elemi pont műveletek .. II-67
 F-83
 POS(0) függvény I-52
 Power-On RESET rutin - INIT ... F-56
 PROGEC - Adat-blokk beolvasás II-65
 F-55,F-94
 PRINT utasítás I-35,F-10
 PRINT USING utasítás I-36,F-2
 PRINT# utasítás .. I-24,I-41,F-2,F-5
 PRINT# USING utasítás I-41
 PRINT\$ utasítás I-36,F-2,F-11
 PRINT\$ USING utasítás I-36
 Program I-1
 - ábrázolás II-1,II-3
 - értelmezés II-2,II-4
 - írás I-13
 - megszakítás I-16,I-23,I-45,II-48
 - mód I-1
 - sor I-2,I-42
 - sor hossza I-14,I-15
 - sor vége I-1,II-2
 - szerkesztés I-13,I-14,I-20
 - tárolás I-44
 - tárolón belüli mozgatása ... II-5
 Programozott hibakezelés I-33
 PRTHND - nyomtató kezelő rutin
 II-52,F-54,F-75

R

RANDOM utasítás I-41
 RDHEAD - Név-blokk keresés/olvasás
 rutin II-64,F-55,F-90
 RDSYN - Blokk szinkronizáló rutin

II-65,F-94

READ utasítás I-24,I-25,I-42
 Real-time óra II-48,F-30
 ?REDO üzenet I-30
 Reláció műveletek I-9,I-12,I-13
 - karakteres adattal csomag . II-24
 - numerikus adattal csomag .. II-24
 REM utasítás I-42
 Rendszerváltozók F-29
 RESET nyomógomb II-6,II-48,F-42
 RESET utasítás I-43,F-2
 RESET rutin II-48,F-54,F-56
 RESTORE utasítás I-23,I-43,F-3
 RESUME utasítás I-33,I-43,F-5
 RESUME 0 utasítás I-43
 RESUME NEXT utasítás I-43
 RETURN billentyű I-1,I-15,I-16
 I-19,II-4,F-11
 RETURN utasítás I-23,I-28
 RG-RETURN without GOSUB hiba .. I-28
 II-18,F-1,F-6
 RIGHTS(x\$,i) függvény I-52
 RND(x) függvény I-53,II-45,F-3
 ROM kapcsok F-36,F-37
 RST belépési pontok F-29
 RUN parancs I-21,I-33,II-6,F-3
 RW-RESUME without error hiba .. I-44
 F-5,F-6

S

SAVE utasítás I-44,F-2
 SAVE SCREEN utasítás I-32,I-44
 SCLK - soros busz órajel . F-41,F-43
 SDATA - soros busz adat .. F-41,F-43
 SET utasítás I-45,F-2
 SETDOT - grafikus pont bekapcsoló
 rutin II-67,F-55,F-83
 SGN(x) függvény I-53,II-46
 SHIFT billentyű I-14,I-16
 SIN(x) függvény I-53,II-44
 Skalár tábla bejegyzések II-7
 SKIP: üzenet I-32
 SN-Syntax Error hiba I-15,I-25
 I-28,I-50,I-52,F-1,F-6
 Sor beolvasó rutin - GLINE ... II-53
 F-54,F-76
 - feldolgozás II-4

- hossz I-14, I-15
 - szám I-2, I-14, I-15, II-2
 - szerkesztés I-14
 - lezárása (SHIFT/+) .. I-15, F-13
 - újratekésztése (SHIFT/+) I-15, F-13
 - törlése I-13, I-14, I-20
 Soros busz figyelmenfelhívás (ATN)
 F-41, F-43
 - csatlakozó F-41, F-43, F-44
 - kiszolgálás kérés (SRQ) F-41, F-43
 - vezérlés F-41
 Speciális billentyűk I-14, I-16
 SQR(x) függvény I-53, II-45, F-3
 SRQ - soros busz kiszolgálás kérés
 F-41, F-43
 ST-String Formula Too Complex hiba
 I-47, F-4, F-6
 STACK I-48, II-14
 - bejegyzések II-15
 - FOR adatcsomag II-16
 - GOSUB adatcsomag II-18
 - kifejezés csomagok II-21, II-23
 - törlése I-23
 STEP kulcsszó I-27
 STOP utasítás .. I-20, I-26, I-45, II-6
 STR\$(x) függvény I-53
 String I-3
 STRINGS(i, j, l, x\$) függvény .. I-54, F-2
 SYSTEM - PRIMO ROM START F-54

SZ

Számábrázolás I-3
 - egész típusú II-29
 - valós típusú II-32
 Számspecifikáció I-36
 Számjegyekarakter I-2
 - konstans I-3, I-24, I-25
 Szerkesztés I-13, I-14, I-20
 Szorzó rutinok ... II-40, II-41, II-43

T

TAB(i) függvény I-54, F-3
 TAN(x) függvény I-54, II-44
 TAPE csatlakozó F-39, F-40, F-42

Tároló felosztás F-27, F-28
 - igény I-6, II-1

Természetes alapú logaritmus (LOG)
 függvény I-51, II-45, F-2
 Területfoglalás gépi kódú rutinok
 számára II-24, II-25, II-26
 TEST utasítás I-45
 TIMER rutin II-48, F-55
 Típus azonosító karakterek I-5, II-6
 - deklaráció I-5, I-25
 - keveredés (TM) hiba I-7, I-28
 I-31, I-41, F-4, F-6
 - kód II-7, II-8, II-9, II-10
 II-17, II-37, F-34
 - konverzió I-7, I-48, I-49
 Tizedes pont I-3, I-4, I-38
 TM-Type Mismatch hiba I-7, I-28
 I-31, I-41, F-4, F-6
 Token II-1, F-45
 Tokenizált forma II-1, II-4
 Tömb I-6, I-26, II-10, F-3
 - újradimenzionálás (DD) hiba I-26
 F-3, F-6
 - változók I-6, F-3
 Törlés (karakter, sor) I-14
 TROFF parancs I-22
 TRON parancs I-22
 TSTDOT - grafikus pont vizsgáló ru-
 tin II-67, F-55, F-84
 Túlcsordulás (OV) hiba I-7, I-9
 I-10, I-22, I-34, I-46, F-3, F-6
 Túlságosan bonyolult karakteres ki-
 fejezés (ST) hiba I-47, F-4, F-6
 Tükör byte F-40

U

UE-User Error hiba I-27, F-5, F-6
 Ugrótábla II-47
 Új karakterek definiálása F-17
 UL-Undefined Line hiba ... I-14, I-21
 I-28, I-29, I-33, I-43, I-44, F-3, F-6
 Upper billentyű I-16
 Utasítások I-22
 - végrehajtása II-5

INDEX



V24 interface F-39,F-40
 VAL(x\$) függvény I-54
 Valós numerikus konstans I-4
 - típusú számbábrázolás II-32
 VARPTR függvény I-55
 Változók I-5
 - nevek I-5
 - táblázatai II-6
 - egész típusú II-6
 - egyszerespontos valós II-7
 - duplapontos valós II-7
 - karakteres II-10
 - indexelt II-10
 - tároló igénye I-6
 Várakozás képkioztásra - VBLANK ru-
 tin II-49,F-55
 Vezérlő billentyűk I-16,F-9
 - karakterek F-9
 Végrehajtási sorrend I-12
 Véletlenszám sorozat I-41
 I-53,II-45

Vízszintes írás (CTR/W) F-13
 - tabulátor (SHIFT/+) II-54
 F-10,F-11

W

WRHEAD - Név-blokk feliró rutin
 II-63,F-55,F-87
 WRREC - Adat-blokk feliró rutin
 II-63,F-55,F-86

X

X^Y hatványérték függvény I-8
 I-13,II-45

Z

Záró-blokk II-55,II-58
 - feliró rutin - ENDREC II-64
 F-55,F-88



**Forgalmazó:
ELEKTROMODUL**

**1132 Budapest, Victor Hugo u. 13-15.
Telefon: 495-340 ● Telex: 22-5154**

**Menedzser:
MTA-SZTAKI COSY MŰSZAKI FEJLESZTŐ LEÁNYVÁLLALAT
BUDAPEST 5.
Pf.: 690
1365**